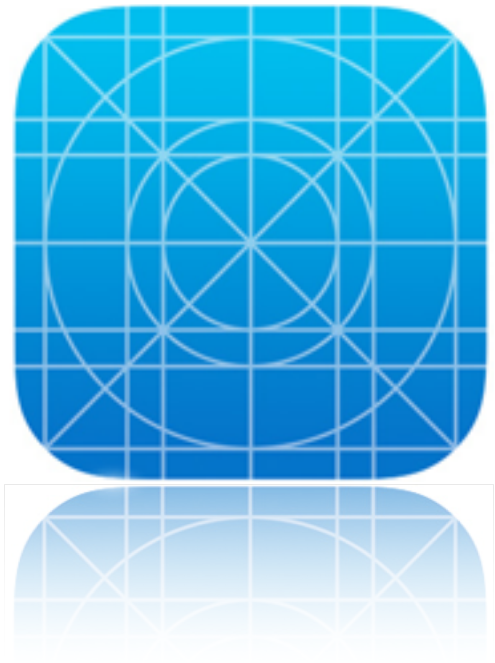
FIB - UPC



Appnalyze

Clasificación de comentarios de aplicaciones móviles

Memoria del Proyecto

Estudiante: SALVADOR MEDINA HERRERA
Presidente: LLUIS PADRO CIRERA
Vocal: ROSENDO REY ORIOL
Secretario: JORDI TURMO BORRÁS
Titulación: Ingeniería Superior Informática
Créditos: 37.5

AGRADECIMIENTOS

Con esta memoria concluyen largos meses de intenso trabajo y reuniones. Días y días de proceso, Gigabytes y Gigabytes de datos. Han aparecido no pocas complicaciones y ha sido necesario repetir pruebas una y otra vez hasta haber podido garantizar el nivel de exigencia requerido para este proyecto. Sin embargo, y pese a todo lo realizado, lo que hay detrás de esta memoria es precisamente el objetivo que ha de tener un PFC: un montón de cosas aprendidas que no hubiese sido posible descubrir de otra manera, ni en la más complicada de las asignaturas.

El hecho de compaginar el desarrollo del PFC con el trabajo a tiempo completo ha sido uno de los mayores retos a los que me he tenido que enfrentar. Esto ha sido a la vez algo muy gratificante, pues considero de enorme importancia para la vida de un ingeniero no conformarse y ser capaz de compaginar trabajo con desarrollos propios.

Me gustaría finalmente agradecer a mi tutor Jordi Turmo Borrás todo el soporte que me ha ofrecido tanto durante la realización de este proyecto como durante mi paso por esta universidad. De especial utilidad formativa han sido las prácticas en el departamento LSI bajo su tutoría, pues me ayudaron a introducirme de lleno en la materia tratada en este proyecto.Con esta memoria concluyen largos meses de intenso trabajo y reuniones. Días y días de proceso, Gigabytes y Gigabytes de datos. Han aparecido no pocas complicaciones y ha sido necesario repetir pruebas una y otra vez hasta haber podido garantizar el nivel de exigencia requerido para este proyecto. Sin embargo, y pese a todo lo realizado, lo que hay detrás de esta memoria es precisamente el objetivo que ha de tener un PFC: un montón de cosas aprendidas que no hubiese sido posible descubrir de otra manera, ni en la más complicada de las asignaturas.

MOTIVACIÓN

No cabe duda de que la forma en que tanto empresas de software y proveedores de servicios como consumidores entienden la informática, internet y las nuevas tecnologías actualmente es sustancialmente diferente a la concepción generalizada de unos pocos años atrás.
Para los consumidores todo parece haberse reducido y diversificado enormemente. Si pensamos en el segmento del hardware, lo que antes se hacía con un único y voluminoso instrumento ahora se hace con diversos aparatos muy especializados.
En cuanto al software, la evolución ha seguido la misma tendencia. Hemos pasado del navegador web y el block de notas a llevar sólo en nuestro teléfono móvil decenas de aplicaciones muy simples con las que realizamos tareas de lo más diversas.

PROBLEMA A SOLUCIONAR

Resultado de esto es que las empresas, que antaño solían basar su actividad en el desarrollo y mantenimiento de una única aplicación, ahora se ven obligadas a producir decenas de aplicaciones para un público cada vez más masivo y diverso. Y quizás una de las mayores dificultades derivadas de este cambio es que los medios que anteriormente se usaban para dar soporte a los clientes han dejado de ser viables.

La comunicación con estos últimos es cada vez menos directa y por tanto es necesario analizar nuevas fuentes de feedback como pueden ser menciones en las redes sociales, reviews, etcétera. Desgraciadamente, realizar este proceso de forma manual es muy costoso y deben buscarse métodos para que pueda ser realizado de forma automática.

OBJETIVOS

Este proyecto tiene por objetivo presentar y analizar las bases de una posible solución al problema anteriormente planteado. Estudiaremos la aplicación técnicas de Machine Learning, en particular de diferentes modelos de Support Vector Machines, para clasificar conjuntos de comentarios de aplicaciones. Estos comentarios serán extraídos de tiendas online de aplicaciones para móviles.

En particular, utilizaremos técnicas de aprendizaje automático supervisado y compararemos diferentes modelos capaces de decidir si cada uno de los comentarios analizados contienen o no oraciones con información subjetiva sobre la aplicación, es decir, si en ellas se valora la aplicación en cuestión. Además, asignaremos cada una de estas valoraciones a cero o más temas generales. Dichos temas serán, como ya se comenta en apartados posteriores, la facilidad de uso, la estabilidad, la funcionalidad, el diseño y el rendimiento. También será parte del abasto de este proyecto la inferencia de la polaridad de dichas valoraciones, es decir, descubrir si se trata de comentarios positivos o negativos.
Además analizaremos las posibilidades de extensión que nos proporcionan estos modelos para aplicaciones de ámbitos diversos. A vista de los resultados decidiremos si es necesario construir modelos de detección diferentes para analizar comentarios de aplicaciones correspondientes a ámbitos diferentes o es posible realizar un único modelo para cualquier tipo de aplicación.

El objetivo final de este proyecto es por tanto estudiar los valores óptimos que deberían tener los clasificadores anteriormente mencionados. Acotaremos este estudio a los clasificadores basados en SVM binarios o multi-clase. Comprobaremos también el efecto que supone la selección de diferentes conjuntos de atributos lingüísticos con los que representar cada documento de entrada y diferentes métodos de reducción del de número de los mismos.

Cabe destacar que una de las fases más importantes de este proyecto es la de generación de los corpus de aprendizaje y evaluación. Deberemos marcar manualmente miles de documentos de opinión siguiendo unos criterios cerrados y estrictos. Adicionalmente deberemos seleccionar y equilibrar estos conjuntos de elementos con el fin de que seamos capaces de obtener los mejores resultados posibles mediante los sistemas de aprendizaje automático que aplicaremos posteriormente.

RESUMEN

A continuación listaremos de forma resumida los diferentes objetivos marcados para este proyecto:

- Profundizar en el estudio de los clasificadores basados en Support Vector Machines y los problemas de clasificación.
- Evaluar la viabilidad de aplicar clasificadores basados en SVM para clasificar las diferentes oraciones de las que se compone un comentario de una aplicación móvil en función de su polaridad y de las características que se valoran en las mismas.
- Compararar las diferencias que podemos encontrar al aplicar diferentes modelos de clasificación muti-clase.
- Afianzar los conocimientos adquiridos durante la carrera en los campos del aprendizaje automático supervisado y del procesamiento del lenguaje natural.
- Sentar las bases necesarias ya sea para una hipotética extensión o estudio posterior o bien para una posible aplicación comercial del estudio realizado en este proyecto.

ESTADO DEL ARTE

EL PROBLEMA DE LAS CLASIFICACIÓN

El problema de la clasificación, especialmente cuando se encuentra ligado a la aplicación de técnicas de inteligencia artificial o aprendizaje automático, es una de las áreas más prolíficas de la imformatica, atendiendo al número de publicaciones y aplicaciones posibles que podemos encontrar.
Este es sin duda un campo de lo más variopinto, pues desde sistemas de ordenación de resultados para páginas web hasta algoritmos de reconocimiento visual son tipos particulares de problemas de clasificación.

No debe resultar extraño por tanto que exista una gran cantidad de algoritmos de clasificación, tanto específicos a un ámbito como generales. Ejemplos de estos algoritmos de uso general son k-nearest Neighbors (k-NN), las redes bayesianas, redes neuronales o las Support Vector Machines (SVM).

CLASIFICACIÓN DE TEXTOS

Debido en gran parte al auge de las redes sociales, en los últimos años han aparecido numerosas herramientas de clasificación o detección de textos así como gran cantidad de estudios y nuevas técnicas que permiten ser mucho más eficaces en esta tarea. De especial importancia ha resultado el auge en el interés de la minería de datos, para la cual el uso de clasificadores automáticos resulta de una gran utilidad.

En particular, las técnicas basadas en SVM están dando lugar a múltiples estudios sobre la clasificación de textos, tal y como muestra la literatura en los últimos años (Bolelli et al. (2007); Bordes et al. (2007); Sun et al. (2007); etcétera). Asimismo, debido al gran interés generado, son diversas las implementaciones y las librerías de código libre que podemos encontrar para las diferentes variantes de este algoritmo.

Pese a que pueda parecer una tecnología novedosa, este tipo de algoritmos llevan ya más de una década siendo estudiados y utilizados en otros ámbitos como pueden ser la detección de patrones (Christopher J. C. Burges (1998)) o el filtrado de correo electrónico no deseado (Harris Drucker & Donghui Wu (1999)), además de muchos otros usos en ámbitos relacionados.

CLASIFICACIÓN DE COMENTARIOS EN LENGUAJE NATURAL

Pese a que el problema de la clasificación comentarios en lenguaje natural representa únicamente un subconjunto de la clasificación de textos. Sin embargo sus características particulares y el enorme interés despertado en los últimos años han propiciado que se realicen gran cantidad de publicaciones e investigaciones centradas en su categorización y clasificación.

La clasificación de comentarios de aplicaciones entra dentro de esta categoría, pues se trata de documentos cortos, muy subjetivos y que a menudo contienen importantes errores sintácticos o gramaticales.

Por suerte, la utilidad de los SVM en estos casos se encuentra ya confirmada por numerosos estudios como Mining Adverse Drug Reaction Signals from Social Media, de la Arizona State University; o Twitter Sentiment Classification using Distant Supervision, de la Stanford University.

CARACTERÍSTICAS CLASIFICADAS

INTRODUCCIÓN

Como ya se comenta en la introducción de este proyecto, el objetivo principal del mismo es estudiar la clasificación de comentarios de aplicaciones móviles atendiendo a la característica o características tratadas en el mismo.

Debido a que el tiempo de desarrollo de este proyecto es limitado, ha sido necesario definir muy claramente cuáles son aquellas características para las cuales se realizaría el estudio. Con el fin de que la mayor parte de los documentos quedasen clasificados en alguna de dichas características, estas deben ser lo más generales posibles.

La decisión de clasificar unas características en lugar de otras puede resultar arbitraria. Sin embargo, si se presta atención a los principales criterios que se plantean en las páginas de reviews profesionales de aplicaciones como “www.theiphoneappreview.com” o “www.pcmag.com”, existe una serie de características que son comentadas en la práctica totalidad de las reviews realizadas: el diseño y simplicidad, la estabilidad, el rendimiento y la utilidad que proporciona la aplicación.

CRITERIOS

El siguiente paso realizado fue la definición de unos criterios objetivos con los que definir cada una de las características. Así, todos los ejemplos seleccionados para la construcción y evaluación de los clasificadores han sido obtenidos siguiendo escrupulosamente los criterios que se listan a continuación:

Diseño

- La oración hace referencia a un aspecto puramente estético de la aplicación. Ejemplos de este concepto podrían ser “La aplicación es bonita” o “Los botones me encantan, son super monos”.
- La oración explica en qué nivel se adapta el diseño de la aplicación a los patrones generales de la plataforma. Esto es, cuánto de coherente es el diseño de la aplicación con el diseño general del sistema operativo y sus componentes básicos. Un ejemplo de este criterio podría ser “Parece una aplicación de Android 2.3” o “Es un calco de la aplicación de iPhone, muy mal…”.
- La declaración dice cómo de bien se adapta el diseño de la aplicación al tamaño y resolución de la pantalla del dispositivo. Por ejemplo “Todo se ve descolocado en mi Galaxy S” o “Las imágenes se ven muy pixeladas”.

Rendimiento

- La oración valora la velocidad de la aplicación a nivel de interfaz y de respuesta al usuario. Ejemplos de este concepto podrían ser “La aplicación va como la seda”, “Se ralentiza cada dos por tres” o “Tarda mucho en abrirse”.
- La oración realiza un comentario relativo al consumo que hace la aplicación de los recursos del dispositivo móvil. Un ejemplo podría ser la oración “La aplicación consume mucha más RAM que antes”.

Funcionalidad

- La oración valora el hecho de que la aplicación proporcione o no proporcione una determinada funcionalidad. Cabe destacar que no se busca descubrir, al menos inicialmente, de qué funcionalidad particular se trata. Un ejemplo de este criterio podría ser “No permite ver las series, sólo son noticias…”.

Estabilidad

- La declaración expresa alguna situación para la cual la aplicación no responde correctamente. Estas respuestas incorrectas pueden ser bajadas de velocidad, pérdidas de responsividad a los diferentes eventos o sensores del dispositivo, cuelgues o bugs en general. Ejemplos de este criterio podrían ser “La aplicación se cierra al pulsar la pestaña de comentarios” o “La aplicación no se abre en mi Xperia Mini”.
- La declaración contiene una valoración sobre el funcionamiento errático de algunas secciones básicas de la aplicación. Un ejemplo de este criterio podría ser “Cuando pulso en el botón de login no hace nada”.
- La declaración demanda a los desarrolladores que un cierto bug técnico de la aplicación sea solucionado o felicita a los mismos por la resolución de alguno de los bugs contenidos previamente a la publicación de una nueva versión de la misma. Una oración que ejemplifica este criterio es “Por fin han arreglado el crash en la pantalla de búsqueda, ya era hora”.

Facilidad de uso

- La declaración valora el grado de intuitividad, accesibilidad o comodidad ya sea para acceder a una cierta funcionalidad particular de la aplicación como en un uso en general. Ejemplo de este criterio puede ser “La aplicación ahora es mucho más intuitiva”.
- La declaración informa sobre la imposibilidad de acceder a alguna de las funcionalidades de la aplicación especificadas en la descripción de la misma siempre que no sea resultado de un error técnico de la misma. Por ejemplo, “No sé como añadir un comentario” cumpliría este criterio.
- La declaración expresa la opinión del usuario sobre la documentación y los tutoriales incluidos o no en la aplicación.

ACLARACIONES FINALES

Cabe recordar que en muchos casos, incluso siguiendo unos criterios objetivos, la clasificación de algunos documentos puede ser ambigua. Esta ambigüedad viene en muchas ocasiones derivada de los errores gramaticales que se encuentran en la misma o del uso de palabras que pueden dar lugar a confusión. En estos casos, los documentos serán marcados con todas las características entre las que se produce la ambigüedad. Un ejemplo de este caso es la oración “La aplicación va a trompicones”, que según este criterio pertenecerá a la vez a la característica “Estabilidad” y “Rendimiento”

CLASIFICACIÓN SEGÚN LA POLARIDAD

INTRODUCCIÓN

El segundo criterio de clasificación considerado para este proyecto es la polaridad o *sentiment*, es decir, determinar si las valoraciones realizadas resultan positivas o negativas.

Según la definición anglosajona de este concepto, podemos entender el concepto de *sentimient* tanto como el pensamiento influenciado o provocado por un sentimiento o una emoción como la manifestación de esta emoción.[1] Estas manifestaciones de subjetividad se pueden encontrar en los textos o comentarios en formas muy diversas y será este concepto el foco de nuestro estudio.

La polaridad puede restringirse también a algún ámbito en concreto. Por ejemplo, un documento puede contener una valoración positiva respecto a un cierto concepto como puede ser el rendimiento de una aplicación y una valoración negativa o incluso no subjetiva sobre otro concepto como podría ser el diseño. En este proyecto abordaremos ambos casos, es decir, estudiaremos la clasificación de la polaridad de las oraciones como conjunto o restringida a una cierta característica valorada.

CRITERIOS

Siendo este un concepto tan subjetivo, podemos encontrar diferentes definiciones para cada uno de los grados de polaridad que pueden ser clasificados. En este proyecto consideraremos cuatro grados posibles. La definición de dichos grados se muestra a continuación:

- Documento no subjetivo. Un documento no subjetivo es aquel que no contiene ninguna valoración subjetiva sobre la aplicación o la característica tratada. Cabe destacar que en el ámbito de los comentarios sobre aplicaciones móviles existen muy pocas oraciones que no contienen ningún elemento que denote subjetividad.
- Documento positivo. Un documento es positivo cuando únicamente contiene valoraciones positivas sobre la aplicación o la característica tratada o cuando la valoración global del mismo resulta positiva.
- Documento neutro o ambiguo. Un documento es neutro cuando contiene tanto valoraciones positivas como negativas sobre la aplicación o característica valorada o bien cuando, debido a la ambigüedad del lenguaje utilizado, no es posible determinar la polaridad.

[1] Fuente: http://www.wordreference.com/definition/sentiment

- Documento negativo. Un documento es negativo cuando únicamente contienen valoraciones negativas sobre la aplicación o característica tratada o cuando la valoración total del mismo resulta negativa.

CONSIDERACIONES ADICIONALES

Cabe recordar que clasificación se realizará a nivel de oración. Esto implica que puede darse el caso de que existan comentarios que globalmente sean negativos pero contengan oraciones en las que solo se realicen valoraciones positivas. Por tanto, estas oraciones serán identificadas como positivas aunque la valoración global del comentario en el cual están contenidas sea negativa. Otra implicación de esto es que no se detectarán los elementos que denotan subjetividad dentro de una oración sino que la oración siempre será valorada como un conjunto.

ORGANIZACIÓN Y SELECCIÓN DE COMENTARIOS

En esta sección se definen las decisiones tomadas para la selección de los criterios que han los grupos de aplicaciones de los cuales obtendremos los documentos con los que construir los diferentes corpus. Además se explica cómo se han seleccionado los comentarios sobre los que finalmente se ha trabajado.

SELECCIÓN DE LA TIENDA DE APLICACIONES

Actualmente existe una gran cantidad de tiendas de aplicaciones disponibles en el mercado de las cuales pueden ser extraídos conjuntos de comentarios. Estas tiendas se pueden dividir en dos grupos fundamentales: las tiendas independientes y los agrupadores de aplicaciones.

Las primeras son aquellas que permiten la descarga y la compra de aplicaciones independiente de otras plataformas. En la mayoría de ocasiones se encuentran preinstaladas en los dispositivos y no pueden ser desactivadas por los medios habituales. Ejemplos de estos tipos de tiendas son “App Store”, “Google Play” o “Amazon Apps”.

El segundo grupo es el de aquellas aplicaciones sitios web que contienen únicamente enlaces a otras tiendas de aplicaciones y no disponen de permisos para la instalación automatizada. Ejemplos de este tipo de tiendas son “AppBrain” o “Mevvy”.

Los dos criterios fundamentales para la selección de la tienda de la que serían extraídos los documentos a ser analizados que fueron decididos son la variedad y número de comentarios en castellano y la facilidad para la extracción de los mismos. Los resultados de la selección se pueden apreciar en la tabla siguiente:

	Dificultad de Extracción	Número de Comentarios	Comentarios Adicionales
App Store	Muy Alta	Muy Elevado	La versión web únicamente muestra 5 comentarios de cada aplicación, lo que dificulta mucho la extracción.
Google Play	Media	Muy Elevado	
Amazon Apps	Media-Baja	Medio	
Blackberry World	Media	Bajo	
Windows Phone Store	Media-Baja	Medio	
Mevvy	Media	Muy Bajo	
AppBrain	Alta	Muy Elevado	Contiene todos los comentarios de Google Play así como los propios de la plataforma pero no se encuentran divididos por idioma, lo que dificulta en gran medida la extracción y clasificación de los mismos.

Como se puede apreciar en la tabla anterior, la opción que mejor equilibra ambos criterios es la tienda de aplicaciones “Google Play”. Por este motivo se decidió que la tienda de aplicaciones seleccionada para la extracción de comentarios fuese la mencionada. En el anexo “Appnalizer Downloader” podrá encontrar más información acerca de la aplicación desarrollada para tal efecto.

SELECCIÓN DE LOS DOMINIOS

La selección de categorías de aplicaciones o dominios en los que se agrupan los comentarios para su análisis es una tarea de gran dificultad. Queríamos que idealmente las categorías fuesen lo suficientemente diferenciadas para que fuese posible apreciar lo más claramente posible las dificultades que supone aplicar un SVM creado a partir de ejemplos de un determinado dominio a un dominio completamente diferente.

Después de analizar la situación se llegó a la conclusión de que el criterio que nos permitiría cumplir con este objetivo es el hecho de que el cliente objetivo sea lo más diferente posible. De este modo, buscando categorías de aplicaciones para un sector demográfico, nivel cultural o intereses diferentes conseguiremos grupos de comentarios lo más diferenciados posible. Aún así, existen infinidad de conjuntos de categorías de aplicaciones que cumplen este criterio, así decidimos añadir un criterio adicional: que en el momento de la selección se encontrase alguna aplicación correspondiente a alguna de estas características en la portada de la web.

Algunas de estas características se listan en la tabla siguiente. Cabe aclarar que los valores dados a cada celda son absolutamente subjetivos (Aunque consensuado en un grupo de tres personas) y no proceden de ningún estudio de mercado. Un análisis más en profundidad se hubiese excedido del abasto de este proyecto y por tanto fue omitido.

	Grupo de Edad	**Nivel Cultural**	**Intereses**	**Comentarios Adicionales**
Organizadores de Tareas	-	Medio-Alto	-	
Reproductores de Música	Joven	Medio-Bajo	Música	
Seguidores de Actividad Deportiva	Joven	-	Deporte	
Videojuegos	Niño/Joven	Medio-Bajo	Videojuegos	
Aplicaciones de Noticias	Adulto	Medio-Alto	Actualidad	
Aplicaciones de Referencia	Joven/Adulto	Alto	Cultura	
Aplicaciones de Mensajería	Joven/Adulto	-	Social	
Launchers	Joven	Medio-Alto	Teconología	
Live Wallpapers	Niño/Joven	Medio-Bajo	-	El número de opiniones en castellano es reducido
Aplicaciones de Compras	Joven/Adulto	-	Moda	

De entre esta lista, decidí tomar las categorías reproductores de música, aplicaciones de noticias y aplicaciones de mensajería siendo las dos primeras destinadas a un sector demográfico diferente y siendo la tercera una categoría en evidente auge en la actualidad y de unas características claramente diferenciadas a las anteriores, pues existe un importante componente social.

SELECCIÓN DE LOS COMENTARIOS

Para garantizar la máxima variabilidad en los comentarios y evitar una sobrecarga del sistema de descarga de comentarios se ha optado por elegir aplicaciones con un número de descargas inferior 50 millones. Se puede comprobar que aplicaciones con un número más elevado de descargas tienden a ser aplicaciones mucho más depuradas y los comentarios suelen ser mayoritariamente positivos y carentes de información adicional. Es por esto que se consideró la idea de seleccionar únicamente aplicaciones con un número medio o bajo de descargas cuya distribución de valoraciones fuese lo más uniforme posible.

En cuanto a los comentarios seleccionados, optamos por tomar los aproximadamente 500 últimos comentarios publicados de cada aplicación. Para aplicaciones con un número de comentarios inferior a este margen se seleccionó la totalidad de los mismos. Esto nos ha permitido garantizar un mínimo de 4 aplicaciones distintas por dominio y un mínimo total de 12 aplicaciones. Esto es de vital importancia pues ayuda a dotar de mayor generalidad a los modelos generados.

APLICACIONES SELECCIONADAS

A continuación se listan las aplicaciones que fueron finalmente seleccionadas y marcadas, junto con el total de comentarios marcados:

ID	Dominio	Comentarios Totales	Comentarios Marcados
org.play.music.free	Reproductores	496	493
com.jrtstudio.AnotherMusicPlayer	Reproductores	3760	659
com.maxmpz.audioplayer	Reproductores	4476	512
another.music.player	Reproductores	615	615
com.doubleTwist.androidPlayer	Reproductores	4093	497
com.gi.elmundo.main	Noticias	852	521
com.vocento.abc	Noticias	284	284
com.elpais.elpais	Noticias	985	533
com.grupozeta.elperiodico	Noticias	493	493
com.direction	Noticias	183	183
com.jb.gosms	Mensajería	4480	524
com.bbm	Mensajería	4480	523
com.glidetalk.glideapp	Mensajería	925	413
org.telegram.messenger	Mensajería	4479	517

METODOLOGÍA

INTRODUCCIÓN

En este apartado se presentará el estudio comparativo realizado en este proyecto. En primer lugar, se expondrán los motivos que han llevado a la decisión de utilizar Support Vector Machines como técnica de clasificación. Posteriormente, se detalla la configuración y características de los experimentos llevados a cabo así como las herramientas utilizadas para tal tarea.
De este modo será posible comparar las diferentes aproximaciones consideradas para la resolución de los problemas de clasificación binaria y multimclase en el ámbito de la clasificación de documentos de opinión en lenguaje natural.

JUSTIFICACIÓN

Las técnicas de clasificación basadas en SVM han dado lugar al desarrollo de múltiples estudios sobre su aplicación ámbitos tan diversos como la clasificación de imágenes, detección de patrones, etcétera. En particular, uno de los ámbitos en los que ha tenido una aceptación más elevada es el de la clasificación de textos.

Esto es así porque las SVM, a diferencia de otras técnicas basadas en modelos de espacio vectorial (VSM), disponen de algunas ventajas muy útiles para las tareas de clasificación de textos, en los cuales el número de dimensiones es generalmente muy elevado (Joachims (1998)) :

- No requieren una selección o reducción de los términos. En el caso de que una clase se distribuya en áreas separadas del espacio vectorial, será la redimensión mediante la función de Kernel la que se encargue de solucionarlo.
- No es necesario realizar un esfuerzo en el ajuste de parámetros en el caso de problemas linealmente separables, pues dispone de su propio método para ello.

Adicionalmente, existe una ventaja adicional en su aplicación en estos ámbitos. Las SVM reducen la necesidad de etiquetar instancias de entrenamiento tanto en escenarios de aprendizaje inductivo como transductivos. Siendo este último el caso de nuestro proyecto

Pese a esto, existen otros algoritmos y estrategias novedosos que también dan muy buenos resultados caracterizando textos de lenguaje natural como las Import Vector Machines (IVM), de salida probabilistica. Sin embargo se ha optado por el uso de SVM debido que se trata de un

algoritmo que ya ha demostrado su eficacia en numerosas aplicaciones. Además existen numerosas librerías y aplicaciones que lo implementan muy refinadas y testeadas (Consultar Anexo).

SUPPOR VECTOR MACHINES (SVM)

En la última década, las SVM se han convertido en una de las técnicas más utilizadas para clasificación automática, debido a los buenos resultados que se han obtenido. Esta técnica se basa en la representación de los documentos en un modelo espacio vectorial, y asume que los documentos de cada clase son separables en dicho espacio. En base a esto, trata de buscar un hiperplano que separe ambas clases. Entre todos los hiperplanos posibles que separan las clases, SVM se queda con aquella que maximiza la distancia entre los documentos de cada clase y el propio hiperplano, denominado margen.

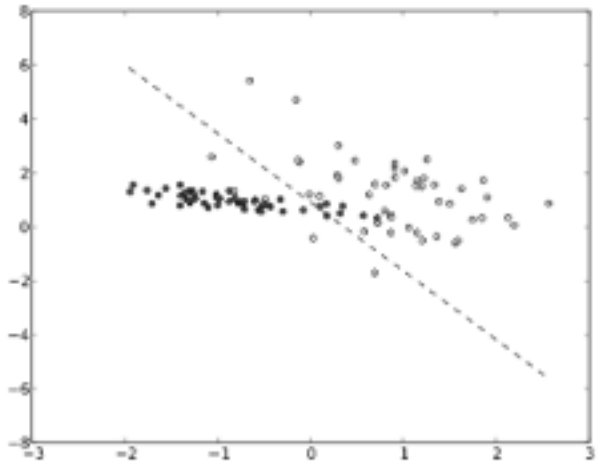


Figura 2.1: Selección del hiperplano que minimiza el margen [www.wikipedia.org]

Dicho hiperplano se define mediante la función $f(x) = w \cdot x + b$. En caso de que el conjunto de datos sea separable, la función anterior obtendrá una correcta clasificación cuando $yi(w \cdot xi + b) > 0 \ \forall i$. Esta relación, además, puede ser reescalada con el fin de obtener su forma canónica, de tal forma que $w \cdot x + b = 1$ defina el hiperplano de margen de una clase, y $w \cdot x + b = -1$ defina las de la otra.

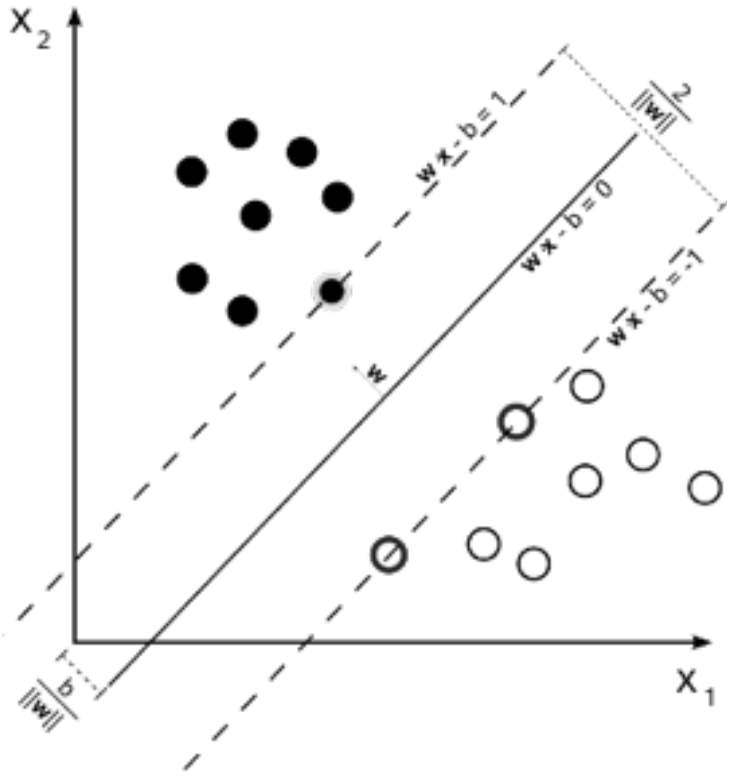


Figura 2.2: Función de clasificación para un SVM
[www.wikipedia.org]

El cálculo de esta función supondría tener en cuenta todos los posible valores para la misma, para después quedarse con los que maximicen los márgenes, lo cual resulta extremadamente difícil de optimizar, sobre todo con un número elevado de ejemplos o cuando los conjuntos de ambas clases no son fácilmente separables. Por esta razón, los clasificadores basados en SVM utilizan la siguiente función de optimización equivalente (Boser et al. (1992); Cortes & Vapnik (1995)) :

$$\arg\min_{\mathbf{w},\xi,b}\left\{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\right\}$$

Donde para todo valor de i, se cumple que:

$$y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

C es el parámetro de penalización, es decir, cuánto se penalizará un ejemplo incorrectamente clasificado. Para valores reducidos de C, un error de clasificación será considerado poco relevante.

Con valores elevados de C, un error de clasificación será muy relevante. Un valor reducido de C suele provocar que el clasificador realice una mala división incluso para el corpus de aprendizaje.

Como se puede deducir de la explicación del parámetro C, la variable Epsilon sub-i es el valor de la distancia entre el hiperplano el vector asociado al ejemplo i.

SOBREAJUSTE EN C

Como ya sabemos, C es el parámetro que define la penalización de un ejemplo mal clasificado del corpus de entrenamiento. Así, cuando el valor de C es elevado, el hiperplano separador tiende a ajustarse lo máximo posible al conjunto de ejemplos del corpus de entrenamiento y el número de ejemplos incorrectamente clasificados para este se hace mínimo.

Este grado de especialización en el corpus de entrenamiento puede ocasionar que su respuesta para un ejemplo que no se encuentre contenido en el mismo pero fuese clasificado correctamente para valores inferiores de C pase a ser clasificado incorrectamente. Este efecto es el conocido como sobreajuste u *overfitting*.

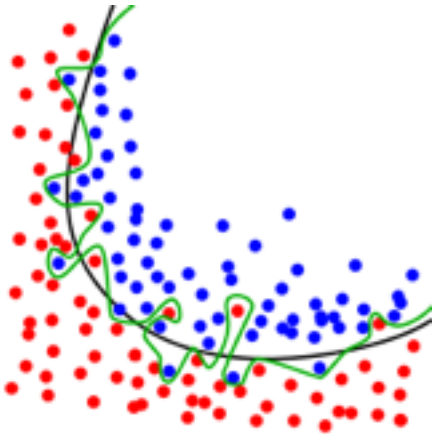


Figura 2.4: Sobreajuste
[www.wikipedia.org]

TIPOS DE KERNEL

A pesar de que la función mencionada anteriormente sí que resulta óptima para obtener el hiperplano que buscamos de forma eficiente, tiene un gran inconveniente: sólo sirve para resolver problemas linealmente separables. Para evitar este problema, se utilizará una función de kernel para redimensionar el espacio de forma que el espacio obtenido sí sea linealmente separable. Posteriormente la redimensión se deshace, de modo que el hiperplano encontrado será transformado al espacio original. Así, obtendremos la función de clasificación que buscamos.

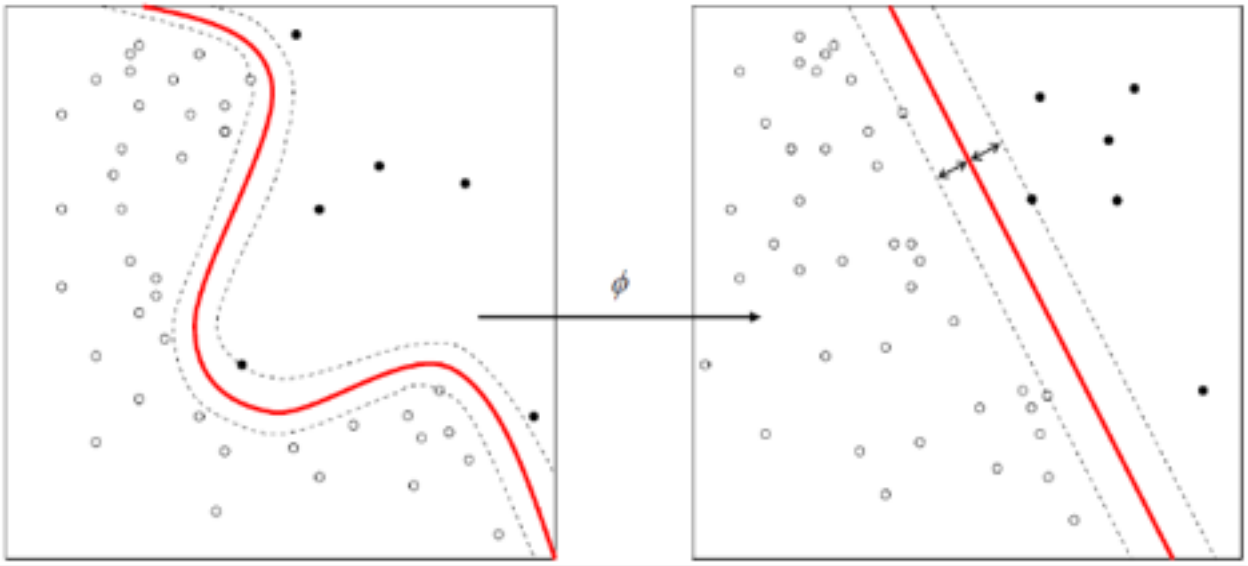


Figura 2.3: Aplicación de una función de kernel sobre el espacio de entrada [www.wikipedia.org]

Las funciones de kernel más utilizadas en la actualidad son las siguientes:

- Lineal: En este caso no se realiza ninguna transformación del espacio.

- Polinómica: En este caso se usa la siguiente función de transformación:

$$k(x_i, x_j) = [\gamma \cdot (x_i \cdot x_j) + r]^d, \gamma > 0$$

  Cuando r toma por valor 0, se denomina homogénea. En caso contrario se tratará de una transformación inhomogénea. Para el caso homogéneo y con d = 0, nos encontramos con el caso lineal.

- Radial Basis Function (RBF). En este caso se utiliza la siguiente función de transformación:

$$k(x_i, x_j) = e^{(-\gamma \cdot \|x_i - x_j\|^2)}, \gamma > 0$$

- Sigmoidea o de Tangente hiperbólica. En este último caso, la función de transformación utilizada es la siguiente:

$$k(x_i, x_j) = tanh(\gamma \cdot x_i^T \cdot x_i + r), \gamma > 0, r < 0$$

Teniendo en consideración el limitado tiempo del que disponemos, hemos decidido centrarnos únicamente en el uso de kernels lineales o polinómicos, ya que se trata de la transformación más simple y diversos estudios ya han demostrado su eficacia en problemas similares.

Obviaremos también el estudio de los kernels polinómicos de inhomogéneos. De lo contrario sería necesario realizar un número muy superior de pruebas, pues tendríamos que obtener los valores óptimos de dos factores adicionales (γ y r).

SVM MULTICLASE

Los SVM son clasificadores naturalmente binarios. Esto puede resultar un problema en algunas aplicaciones, como por ejemplo la detección de la polaridad de los comentarios que hemos tratado en este proyecto. La polaridad en el modo que la hemos definido en este documento no es binaria, sino que puede tener 4 valores diferentes: no subjetivo, positivo, negativo y neutral (positivo y negativo a la vez).
Para solventar esta limitación surgieron nuevos métodos que pudieran resolver sistemas multiclase. Como una primera aproximación directa, Weston & Watkins (1998) propusieron una modificación de la función de optimización, teniendo en cuenta todas las clases:

$$\min \frac{1}{2} \sum_{m=1}^{n} ||w_m||^2 + C \sum_{i=1}^{l} \sum_{m \neq y_i} \xi_i^m$$

$$w_{y_i} \cdot x_i + b_{y_i} \geq w_m \cdot x_i + b_m + 2 - \xi_i^m, \xi_i^m \geq 0$$

Además se propusieron diversas técnicas de aproximación a SVM multiclase a partir de la combinación de múltiples SVM binarios (Hsu & Lin (2001), gozando de una gran popularidad en la actualidad. Estos métodos se basan en dos acercamientos: crear SVM que distingan una a una todas las parejas de clases (one-versus-one) o bien crear SVM que distingan una clase de todas las demás (one-versus-all).

Para el caso de one-versus-all, el criterio utilizado es el de asignar cada elemento a aquella clase cuyo hiperplano separador se encuentre a la máxima distancia del mismo (winner-takes-all). Para el caso one-versus-one, para cada elemento se hace un recuento de las diferentes asignaciones para cada uno de los hiperplanos separadores generados y se asigna a aquella clase a la que haya sido asignado en más ocasiones por los diferentes SVMs binarios (max-wins).

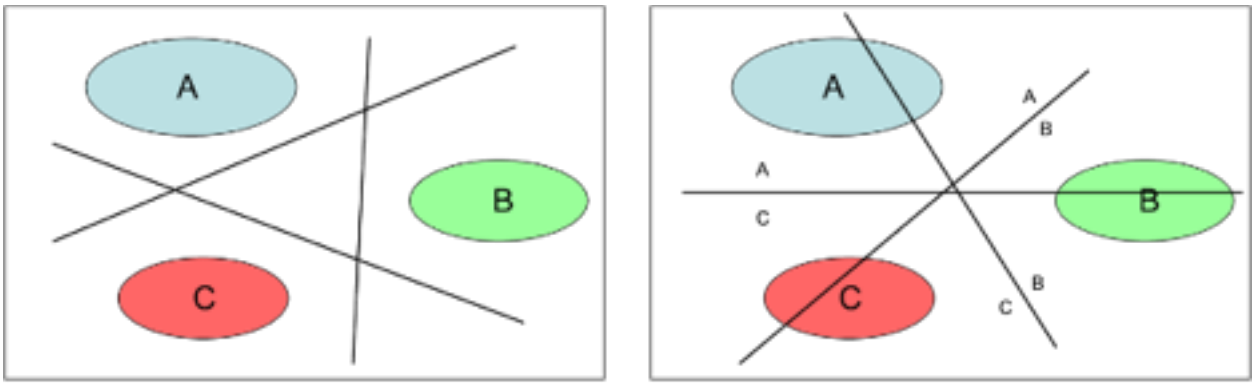


Figura 2.4: A la izquierda, representación de la estrategia one-versus-all. A la derecha, representación de la estrategia one-versus-one.
[courses.media.mit.edu]

En este proyecto se ha utilizado el algoritmo de clasificación muti-clase por defecto ofrecido por la librería LibSVM así como una estrategia de combinación de SVM binarios inspirada en las dos estrategias anteriormente descritas. Los detalles de esta estrategia se describen en el siguiente apartado.

ESTRATEGIA DE COMBINACIÓN DE APPNALYZE

Como ya se ha comentado en apartados anteriores, en este proyecto se ha optado por utilizar un clasificador multi-clase únicamente para discernir la polaridad de los documentos. Los posibles valores de polaridad son no-subjetivo, positivo, negativo y neutro (o ambiguo).

El valor neutro se trata de una combinación de las polaridades negativa y positiva, es decir, aquellos comentarios que cumplen ser positivos y negativos a la vez. También podemos entender esta polaridad como aquellos documentos que son subjetivos pero no se pueden considerar ni negativos ni positivos.

Así, si pensamos en los subconjuntos posibles, diseñando un clasificador positivo+neutro y uno negativo+neutro y obteniendo la intersección de los documentos clasificados, podremos obtener un clasificador para la categoría neutro. Otra opción posible relacionada con la segunda descripción del conjunto neutro dada y equivalente a la anterior sería generar un clasificador positivo, otro negativo y un tercero que detecte si el documento es o no subjetivo y obtener la la diferencia entre los conjuntos clasificados como no-positivos y no-negativos que se encuentran dentro del conjunto de documentos subjetivos.

VALORACIÓN DE LAS SOLUCIONES

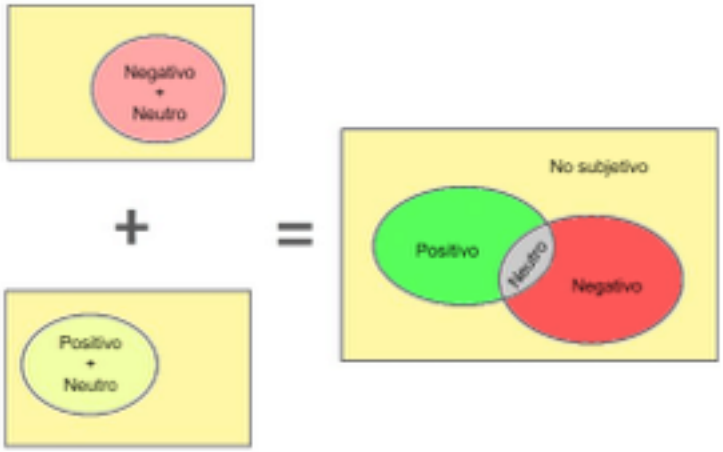


Figura 2.5: Obtención del conjunto de documentos de subjetividad neutra a partir de los conjuntos de subjetividad negativa y neutra y subjetividad positiva y neutra.

Los valores más utilizados para el cálculo de la efectividad de un sistema de clasificación son la precisión y la cobertura. En muchas ocasiones estos dos valores son unificados en la medida . Además de estos tres, en algunos problemas se utilizan otros valores como pueden ser la exactitud o el error.

La precisión se define mediante la siguiente ecuación:

$$precisión = \frac{positivos_{correctos}}{positivos_{correctos} + positivos_{incorrectos}}$$

Guardando estrecha relación con la precisión se define el valor de la exactitud o accuracy y el error como:

$$exactitud = \frac{positivos_{correctos} + negativos_{correctos}}{positivos_{incorrectos} + positivos_{incorrectos} + negativos_{correctos} + negativos_{incorrectos}}$$

$$error = 1 - exactitud$$

La cobertura o recall se define mediante la siguiente ecuación:

$$cobertura = \frac{positivos_{correctos}}{positivos_{correctos} + negativos_{incorrectos}}$$

Es importante destacar además que el cálculo de estos valores se puede calcular de dos maneras diferentes: mediante microaveraging o mediante macroaveraging. En el primer caso, los cálculos se basan en la suma de los valores individuales. En el segundo, los cálculos se basan en la suma local de los valores para cada categoría, obteniendo después la media total.

El cálculo de la Fß se basa en la combinación de estas dos medidas asignando pesos a cada una de ellas mediante la siguiente ecuación:

$$F_{\beta} = \frac{(\beta^2 + 1) \cdot precisión \cdot recall}{\beta^2 \cdot precisión + recall}$$

El caso más habitual suele ser F1, cuando se asigna el valor 1 a la variable ß. Como se puede comprobar, el esta función se maximiza cuando la precisión y el recall tienen un valor similar y elevado.

VALORACIÓN DE LAS SOLUCIONES PARA EL PROYECTO APPNALYZE

Para el ámbito de este proyecto hemos decidido centrarnos en la optimización del parámetro F1, aunque también se han considerado otros parámetros como la exactitud para alguna de las pruebas realizadas.

Resulta también necesario definir el valor mínimo a partir del cual consideraremos correctas las soluciones obtenidas. Los valores típicos para este tipo de proyectos suelen encontrarse entre 0.7 y 0.8. Para este proyecto utilizaremos el valor mínimo de dicho intervalo: 0.7. La decisión de considerar un valor tan permisivo viene derivada de que también serán aplicadas otras restricciones adicionales al valor de F, comentadas en apartados posteriores de este documento.

JUSTIFICACIÓN

Usar el valor de Fß como criterio de evaluación de las soluciones obtenidas puede llegar a ser muy comprometido:

Por un lado, el valor Fß valora únicamente las soluciones para una clase determinada, por tanto para un clasificador binario dado tendremos un valor de Fß para una de las clases y un valor de Fß diferente para la otra. Además, esta medida es una ponderación de otras dos y la elección de un valor particular de es difícilmente justificable.

Por otro lado, otras medidas muy utilizadas en el ámbito del aprendizaje automático como la exactitud parecen generalmente más adecuadas para sistemas en los que se desea proporcionar una respuesta lo más correcta posible y resultan más fáciles de explicar a personas no familiarizadas con el ámbito de la investigación.

Así, aunque la exactitud parezca el valor más óptimo a optimizar para el problema presentado en este proyecto, existen dos razones fundamentales que llevan inequívocamente a descartarlo:

En primer lugar, el número total de documentos es elevado y por tanto el denominador de la función de exactitud es muy grande. Esto ocasiona que en muchos casos, las diferencias en los resultados no sean significativas. Esta problemática no se da únicamente para este problema y ya fue expuesta por Yang (1999).

En segundo lugar, debido a que en la mayoría de pruebas a realizar los corpus no son equilibrados, es decir, el número de documentos de una clase es muy superior al del resto; el valor de la exactitud tenderá a igualar al valor de la precisión para dicha clase. Es decir, si consideramos el caso en el que existe un 95% ejemplos que pertenecen a una clase determinada, el valor de exactitud de un clasificador que asigna toda su entrada a dicha clase será de 0.95, incluso aunque la precisión y la cobertura de la otra clase considerada sea 0.

Otra forma de justificar esta decisión la podemos obtener si reflexionamos sobre el funcionamiento que esperaría un usuario sobre una plataforma de clasificación de comentarios basada en los clasificadores estudiados en este proyecto.
En este escenario, un usuario esperaría que al filtrar los comentarios por un tema, se le mostrasen como resultado el mayor número posible de comentarios que tratan del mismo, es decir, que la cobertura fuese la mayor posible. Además de esto, sería de esperar que entre los resultados mostrados hubiese el mínimo número de comentarios en los cuales no se habla del tema filtrado, es decir, que la precisión sea muy elevada. Vemos así que ambos factores son muy importantes para problema que estamos tratando y confirmamos por tanto que el valor de Fß es el factor de calidad más adecuado.

PONDERACIÓN DE LOS VALORES DE F1

En apartado anterior se comenta una complicación adicional derivada del uso del valor F1: es posible obtener valores de F1 diferentes para cada una de las clases distinguidas por el clasificador. Existen dos posibles soluciones a este problema:

- Definir una clase principal sobre la cual se realizará el cálculo de F1. En este caso, sólo se considera el valor de F1 para una de las clases discernidas. Esta estrategia es útil para problemas binarios en los cuales se desea detectar un cierto conjunto de elementos. Para problemas multiclase esta estrategia no resulta de utilidad.
- Ponderar los valores de F1 obtenidos para cada una de las clases. Dependiendo de la naturaleza del problema puede ser conveniente el uso de un algoritmo de ponderación u otro. Como se puede comprobar, la estrategia anterior es un caso particular de ponderación aritmética.

PONDERACIÓN DE F1 EN EL PROYECTO APPNALYZE

Considerando la justificación presentada en el punto anterior sobre la elección de Fß basándonos en la respuesta que esperaría un usuario de una aplicación de clasificación de comentarios podemos realizar otra observación importante: para la mayoría de escenarios, sólo el factor de calidad Fß aplicado a una de las clases resulta relevante. Así, parece lógico asumir que que la estrategia de ponderación del valor de Fß más adecuada es la definición de una clase principal.

Sin embargo, la elección de este criterio conlleva un compromiso importante: es necesario seleccionar una clase para ser usada como principal. Esta selección puede ser bastante ambigua, pues varía para cada problema particular. Para evitar esta ambigüedad hemos optado por

considerar siempre el caso peor y seleccionar como principal aquella clase para la que se obtiene un valor mínimo de F1. Así, la ecuación que utilizaremos para el cálculo del valor de F1 en cada punto será:

$$F_1 = min\left\{F_1|_{clase\ 0}\ ,\ F_1|_{clase\ 1}\right\}$$

En el caso de los clasificadores multiclase, este criterio pasa a ser demasiado restrictivo. Como ya se comenta en apartado anterior, la estrategia de elección de una clase principal no resulta un criterio adecuado para clasificadores multiclase, pues ignora los valores obtenidos para el resto de clases. Por esta razón, para las pruebas de clasificación multiclase ponderaremos los valores de F1 obtenidos para cada una de las clases mediante una media aritmética.

K-FOLD CROSS-VALIDATION

Debido a que el número de ejemplos del que disponemos es relativamente reducido, sobre todo para algunas de las características que se analizan, la varianza de los valores de calidad obtenidos en una única realización puede llegar a ser muy elevada. Para evitar el efecto de esta incertidumbre se decidió hacer uso de la técnica de valiadación cruzada k-fold. Esto nos ha permitido además aprovechar todos los documentos de ejemplo de los que diponíamos tanto para entrenar como para evaluar los clasificadores generados.

La técnica de k-fold cross-validation se basa en el particionado del conjunto de ejemplos en k subconjuntos. Hecho esto, se realizan k pruebas utilizando para cada una de ellas uno de los subconjuntos como corpus de test y los k-1 subconjuntos restantes como corpus de aprendizaje.

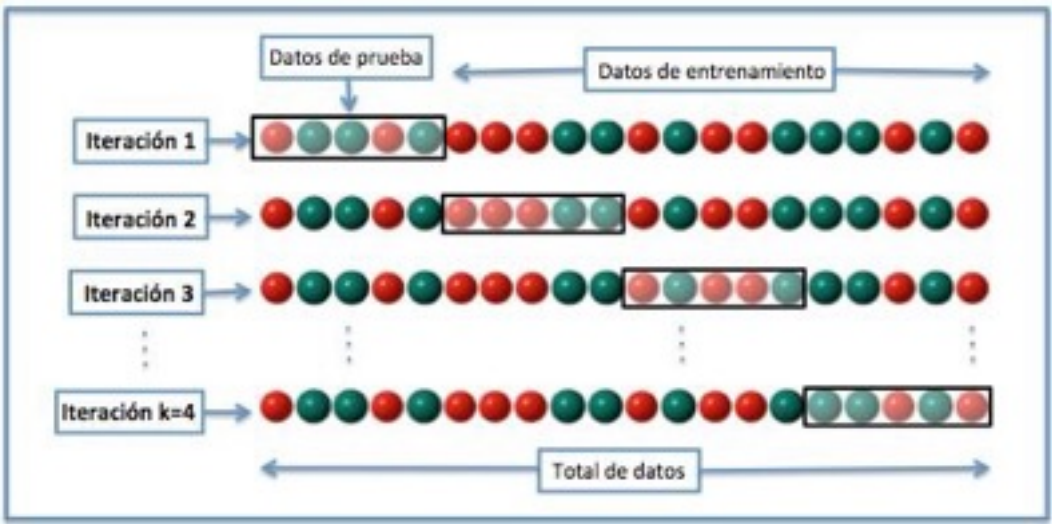


Figura 2.6: K-Fold Cross-validation
[www.wikipedia.org]

En general, los valores típicos que se suelen asignar a la constante k suelen oscilar entre 2 y 10. Para evitar el peligro de o bien no disponer de suficientes ejemplos para la construcción de los clasificadores o bien no disponer de unos corpus de test con un número de documentos suficientemente representativo, hemos decidido tomar un valor de k igual a 5
Existen otras estrategias de validación cruzada muy utilizados como es el caso de la validación cruzada dejando uno fuera. Este tipo de validación cruzada se puede entender como un caso especial de validación cruzada k-fold en la cual k es igual al número total de documentos. Este tipo de validación cruzada, pese a ser muy útil en otro tipo de problemas, no es viable para el caso que estamos tratando si tenemos en cuenta la relación entre el número de ejemplos y el tiempo medio de cálculo de cada realización.

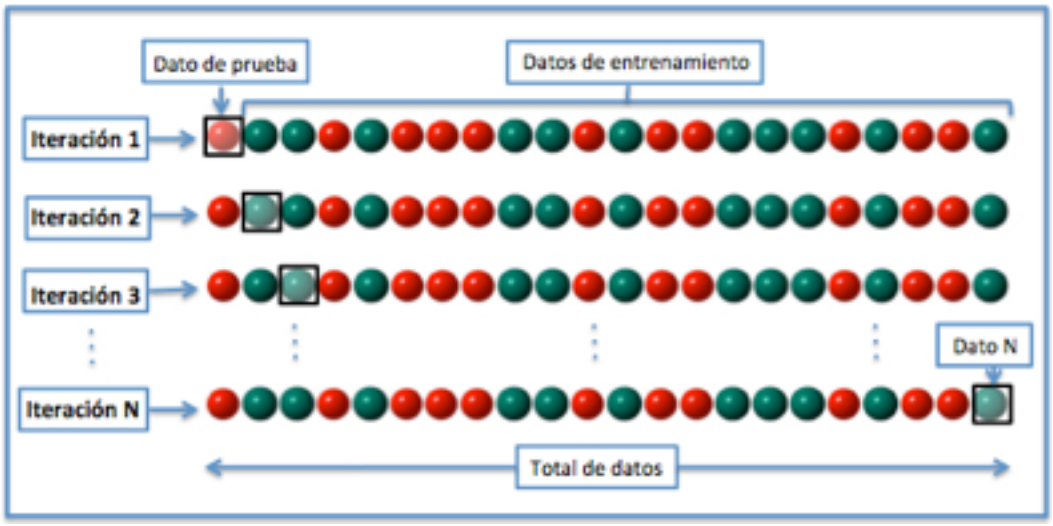


Figura 2.7: Leave-one-out Cross-validation
[www.wikipedia.org]

Con el objetivo de que todas las particiones contengan una proporción equivalente de ejemplos de cada aplicación y cada dominio, un paso previo al participando es la aleatorización de los ejemplos. Esto es, reordenar el conjunto de los ejemplos de manera aleatoria antes de realizar las particiones. Esto nos ayudará también a evitar que nuestros modelos generados puedan tener una cierta dependencia con la fecha de publicación, pues este es el orden inicial con el que son extraídos los documentos.

Cabe aclarar que no todas las las pruebas y análisis que realizaremos requieren el particionado de los ejemplos. En particular, todas las pruebas de aplicación de modelos aprendidos con ejemplos de un conjunto de dominios a ejemplos de un dominio diferente se realizan con todos los ejemplos, ya que el particionado de los mismos ya es implícito a la prueba y por tanto carece de sentido realizar un particionado adicional.

PONDERACIÓN DE LOS RESULTADOS PARA K-FOLD CROSS-VALIDATION

Como ya se explica en el apartado anterior, utilizando la técnica de validación cruzada k-fold se realizarán k evaluaciones diferentes para cada una mima prueba. Esto nos proporcionará por tanto k valores para los criterios de calidad que, cuando el número de ejemplos tiende a infinito, deben tender a un mismo valor. Sin embargo esto no tiene por que ocurrir en un caso real, en el que el número de ejemplos es limitado. La medida calculada presentará por tanto un cierto desplazamiento respecto al valor esperado debido al efecto de la varianza.

Para reducir la varianza de la media que se quiere calcular y obtener un valor más cercano real, el procedimiento habitual consiste en realizar la media de todos los valores obtenidos para cada una de las realizaciones. De este modo el cálculo del valor F1 de resulta el siguiente:

$$F_1 = \frac{\sum_{0}^{k-1} F_1 |_{realización\ i}}{k}$$

REDUCCIÓN DE LOS EJEMPLOS DUPLICADOS

Siendo una parte importante de los comentarios de aplicaciones conjuntos reducidos de palabras, es muy probable que un considerable número de ejemplos se encuentren repetidos. En los sistemas de aprendizaje automático como es este, no tiene por que ser negativo que existan comentarios duplicados, es por esto que se ha decidido no descartar dichos documentos.

Además, en nuestro caso particular esta duplicación puede ser necesaria ya que puesto que la unidad básica usada en nuestro sistema es la oración, puede darse el caso de que haya dos oraciones con exactamente los mismos componentes pero que debido al contexto su polaridad o sus atributos asociados pueden ser diferentes.

Adicionalmente aprovecharemos esta duplicación, como se comenta más adelante, para analizar el efecto de la misma en la calidad de los clasificadores generados. Entenderemos que un ejemplo es duplicado para un atributo en particular cuando, una vez parametrizado, tanto los valores de su parametrización como su polaridad son iguales a los de algún otro ejemplo tomado.

PARAMETRIZACIÓN DE LOS COMENTARIOS

Como ya sabemos, los SVM son máquinas cuya finalidad principal es la clasificación de puntos o vertores contenidos en espacios de múltiples dimensiones. Como se desprende de esta escueta definición, estos autómatas son únicamente clasificadores numéricos y por tanto no son capaces de procesar directamente textos o imágenes por sí solos.

Para que un SVM sea usable en el ámbito del procesamiento del lenguaje natural, debemos llevar a cabo un proceso de parametrización del texto de modo que podamos expresarlo por medio de un conjunto ordenado y finito de valores numéricos. La calidad de los clasificadores generados posteriormente así como su complejidad dependerá en gran medida de los criterios y variables de parametrización elegidos.

Existen gran cantidad de criterios de parametrización de oraciones posibles, como puede ser la posición relativa o absoluta de las palabras, sus categorías gramaticales o sintácticas, etcétera; así como la combinación de varios de los mismos. Es necesario tener en mente que criterios muy generalistas pueden ocasionar una perdida importante de información y criterios muy específicos pueden propiciar un innecesario aumento en la complejidad de los modelos.

No obstante, uno de los criterios más simples pero a la vez uno de los que mejores resultados suele presentar es la morfología, es decir, el hecho de que aparezca una determinada palabra. Este criterio ha visto demostrada su eficacia en numerosas publicaciones científicas y es por eso que hemos decidido usarlo como criterio principal de parametrización de nuestro sistema. Sería interesante comprobar las diferencias que se pueden apreciar cuando es combinado este criterio con otros adicionales como puede ser la categoría gramatical. Sin embargo, por motivos de tiempo, este análisis queda fuera del abasto de este proyecto.

Otra aclaración importante es que el valor de cada uno de los parámetros representará el número de repeticiones de una determinada palabra dentro de un texto. Así nuestro espacio vectorial tomará únicamente valores discretos positivos, es decir no se encontrará normalizado.

REDUCCIÓN DEL NÚMERO DE PARÁMETROS

Para evitar generar un número infinito de parámetros (ya que según el anteriormente mencionado criterio a cada conjunto ordenado de caracteres se le asignará un parámetro diferente), sólo consideraremos válidos los casos que se dan en el corpus de aprendizaje y el resto de palabras posibles se agruparán en el parámetro “otros”.

Además de esto, será realizado un pre-filtrado de los textos con el fin de garantizar que no se añadan parámetros redundantes. En particular, todos los caracteres serán convertidos a minúscula y serán eliminados los acentos y caracteres no alfanuméricos con excepción de los signos de puntuación.

Para algunas pruebas se utilizarán conjuntos de parámetros simplificados, es decir, no incluyendo algunos de ellos. El único criterio de selección que se utilizará en dichos casos es el de la frecuencia de aparición del token determinado en el conjunto de ejemplos. El estudio de otros criterios de selección sería de gran interés, pero escapa del abasto de este proyecto y será dejado para posibles publicaciones posteriores.

Experimentos realizados

INTRODUCCIÓN

En este apartado se detallarán los diferentes experimentos realizados durante este proyecto. Cabe aclarar que parte de los experimentos realizados han sido descartados u omitidos de esta memoria ya sea porque son equivalentes a alguna otra de las pruebas ya detalladas o bien porque no se ha conseguido llegar a una conclusión relevante sobre el resultado de los mismos.

Como ya se menciona en los anexos, los experimentos han sido llevados a cabo utilizando un ordenador personal, con las limitaciones en tiempo de procesado o memoria consumida que esto conlleva. Esta segunda limitación ha sido especialmente limitante, ya que algunas pruebas como las de la generación de SVM sin realizar una reducción en el número de dimensiones o el número de ejemplos no han podido realizarse para todos los documentos marcados, sino únicamente para un subconjunto de los mismos. Las decisiones tomadas a la hora de limitar alguna de las pruebas realizadas se encuentran debidamente detalladas en la descripción de la misma.

Los archivos de configuración utilizados para cada una de las pruebas realizadas se encuentran en el repositorio habilitado a tal efecto. En este apartado no se detallan análisis exhaustivos sobre los tiempos de procesado de cada una de las pruebas sino algunas consideraciones puntuales. Si lo desea, puede consultar los tiempos de cálculo de cada una de las pruebas en el anteriormente mencionado repositorio.

PASOS PREVIOS

Como paso previo a la realización de los experimentos que se detallan a continuación, fueron realizadas una serie de pruebas con valores extremos para los parámetros C y grado del kernel, con el fin de encontrar los rangos más adecuados para las medidas representadas.

En particular, se construyeron y evaluaron SVM para los casos binarios “subjetivo / no subjetivo” y “positivo / no positivo”. Los grados del kernel evaluados fueron 1, 3 , 5 y 8. Para el parámetro C se utilizaron como valores posibles todas las potencias de 8 dentro del intervalo [8^-3, 8^3].

De dichas pruebas fue posible concluir que no se obtiene una mejoría apreciable en el factor de calidad para los kernels de grado superior a 3. Por este motivo decidimos analizar los resultados únicamente para los kernels de grado 1, 2 y 3. En cuanto al valor de C, decidimos utilizar finalmente el rango [4^-4, 4^4], en incrementos de potencias de 4, con el fin de proporcionar una mayor resolución en las medidas realizadas.

CLASIFICACIÓN SEGÚN LA SUBJETIVIDAD

INTRODUCCIÓN

Antes de centrarnos en la detección de la polaridad de un documento de opinión, resulta lógico preguntarnos si es posible discernir entre un documento de opinión y uno que no lo es. Determinar si es posible detectar aquellos documentos que contienen alguna valoración subjetiva se convierte por tanto en el primer experimento de este proyecto.

Los clasificadores que analizaremos en este apartado son binarios, pues un documento sólo puede pertenecer a las clases “subjetivo” o “no subjetivo”. Únicamente serán considerados los kernels lineal o polinomial homogéneo.

DISTRIBUCIÓN DE LOS EJEMPLOS

Debido a que los documentos de entrada son las oraciones en las que se subdivide un comentario de la tienda de aplicaciones Google Play, la probabilidad de que estas contengan algún elemento subjetivo en las mismas es elevado. En particular, 11966 de los 13455 documentos contienen algún elemento subjetivo. O lo que es lo mismo, sólo un 11% de los documentos son no subjetivos.

CONSIDERACIONES ADICIONALES

Para casos con una distribución tan poco equilibrada como es este, resulta de especial interés estudiar cómo se comportaría un sistema en el cual pudiésemos tener una población más equilibrada. Es por esto que además analizaremos los resultados obtenidos por un clasificador SVM para este mencionado caso.

Es también de especial interés en esta prueba la comparación con los resultados obtenidos respecto al clasificador “tonto”, es decir, cuando se tiene un clasificador siempre asigna la clase “subjetivo” a los documentos de entrada.

PRUEBA 1 : CASO GENERAL

La primera parte de este experimento consiste en determinar si es posible construir un clasificador SVM que sea capaz de dividir aquellos comentarios que contienen elementos subjetivos de los que no los contienen.

Hipótesis:

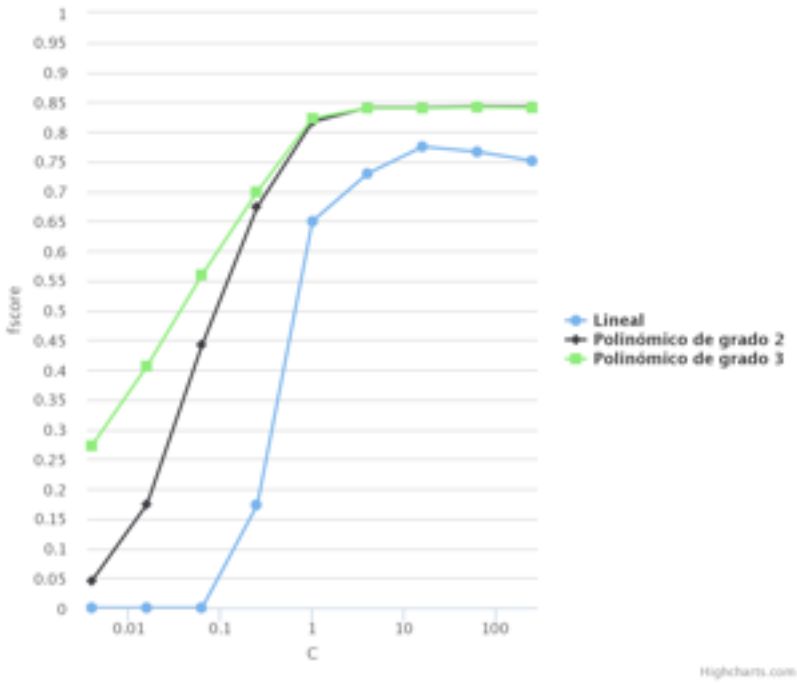
Si no se realiza reducción en el número de características evaluadas, es posible encontrar un clasificador basado en SVM que sea capaz de separar documentos subjetivos de aquellos que no lo son para el factor de calidad mínimo de referencia planteado en este proyecto.

Características del test realizado:

Por limitaciones en la capacidad de procesado del equipo de pruebas utilizado, para esta prueba se utilizará un subconjunto del total de ejemplos, en particular, únicamente serán utilizados los comentarios correspondientes a dos de los tres dominios de aplicaciones estudiados.
Podemos suponer que esto no supondrá una desviación muy elevada respecto los factores de calidad que se obtendrían si la prueba fuese realizada con todos los documentos marcados, pues si bien el número de ejemplos es menor, continúa siendo considerable.

Resultados obtenidos:

A continuación se muestran los resultados obtenidos para la prueba realizada:



Conclusiones y observaciones:

A la vista de los resultados queda comprobada la veracidad de la hipótesis planteada en esta prueba. Confirmamos por tanto que es posible conseguir valores de F superiores a los los límites inferiores definidos para este proyecto para cualquiera de los kernels utilizados.

También podemos comprobar que la respuesta de los kernels de grado 2 y grado 3 es similar para valores de C superiores a la unidad, para los cuales F se hace máximo. El kernel lineal por contra presenta peores resultados que los dos anteriormente mencionados para cualquier valor de C.

PRUEBA 2 : REDUCCIÓN DEL NÚMERO DE CARACTERÍSTICAS

En la primera parte de este experimento se ha comprobado que es posible construir un clasificador de documentos subjetivos y no subjetivos utilizando todas las características obtenidas según la estrategia de parametrización de las oraciones planteada para este proyecto.
A la vista de los resultados parece lógico suponer que existen una serie de características[2] comunes a los documentos subjetivos o no subjetivos que permiten identificarlos. Siguiendo con esta suposición resulta lógico pensar que estas características dispondrán de una frecuencia elevada en el conjunto de documentos.

Hipótesis 1:

Es posible construir un clasificador de documentos atendiendo a si contienen o no elementos subjetivos utilizando únicamente un subconjunto de las características consideradas, sin que esto repercuta en una merma considerable de la calidad respecto a un clasificador para el cual no se realiza tal reducción.

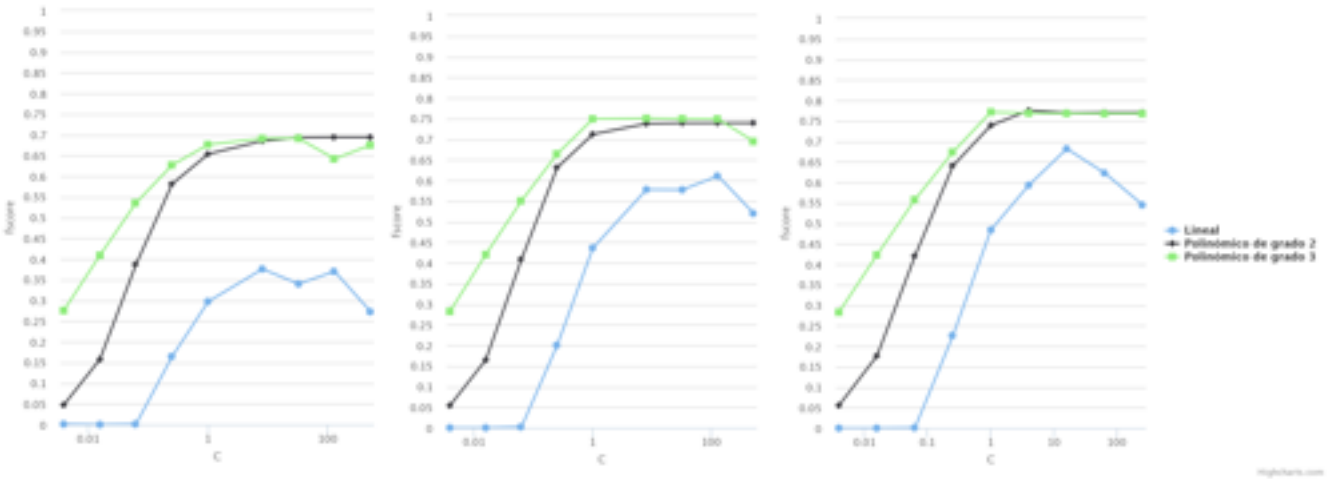
Características del test realizado:

Para la realización de esta prueba se utilizará el total de documentos marcados, pues debido a la reducción en el número de dimensiones del kernel, la capacidad de procesado del equipo de pruebas no ha resultado un factor limitante.
Para empezar, se comprobarán los valores de los criterios de calidad cuando se limita el número de características a las 1000 , 2000 y 3000 características más frecuentes. Esto representa un 11.5%, un 23 % y un 34.5 % del total respectivamente. Cabe aclarar también que ha sido añadida una característica adicional a todos los conjuntos denominada “otros”, que agrupa todas aquellas no seleccionadas.

[2] “Característica” es el término con el que se definen cada una de las posibles parametrizaciones asociadas a cada las dimensiones del espacio vectorial utilizadas por el SVM

Resultados obtenidos:

A continuación se muestran las gráficas de F en función del factor C para la prueba realizada. Las gráficas se presentan de izquierda a derecha para las limitaciones a 1000, 2000 y 3000 características respectivamente.



De cara a las conclusiones resulta también interesante saber que los tiempos de cálculo para cada una de las gráficas han sido de 5 horas y 25 minutos para el primer caso, 6 horas y 5 minutos para el segundo y 6 horas y 55 segundos para el tercero.

Conclusiones y observaciones:

Comprobamos que de nuevo se cumple la hipótesis planteada en esta prueba, pues obtenemos valores de F pico de 0.694, 0.751 y 0.772 respectivamente. Cabe puntualizar que el primero de estos valores es 0.006 inferior al límite establecido para el factor de calidad en este proyecto y por tanto no cumple estrictamente este criterio. Sin embargo, es de esperar que con un ligero aumento en número de parámetros este valor supere el umbral planteado y por tanto tomaremos este valor como válido.

Observamos también, atendiendo a la evolución del valor pico de F, que el incremento en el mismo no es lineal con el número de documentos. Así, sería de esperar que la ganancia obtenida al aumentar el número de características por encima de 3000 resultaría muy poco determinante en el valor del F.

Vemos que el mayor equilibrio entre coste computacional y calidad se obtiene para el caso de la limitación a 2000 características. Por esta razón se utilizará este valor para las pruebas sucesivas.

Hipótesis 2:

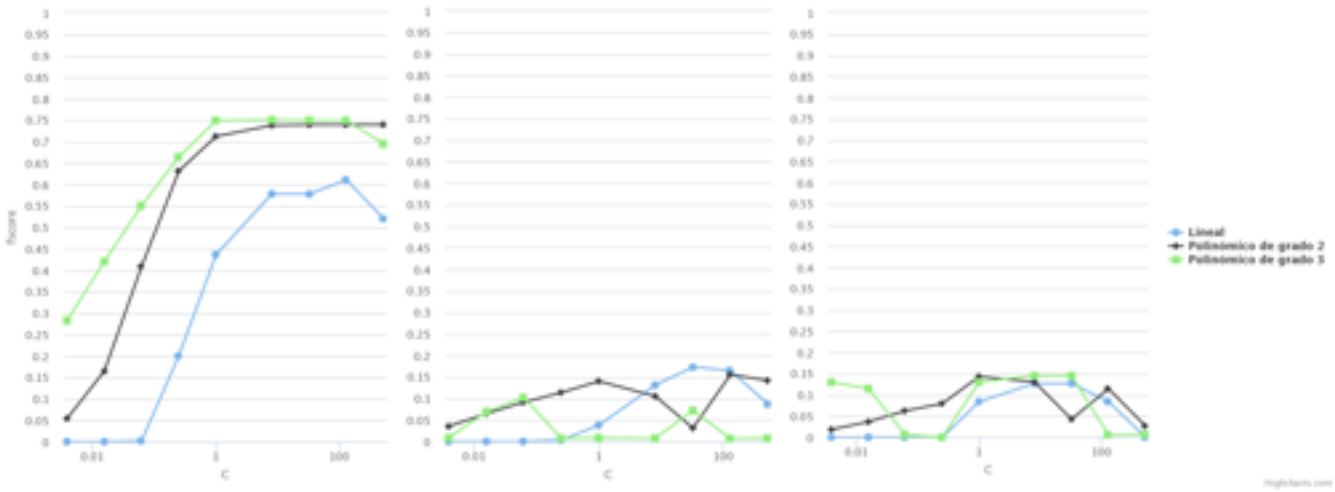
Las características más determinantes para construir un clasificador de documentos en función de su subjetividad son aquellas con mayor frecuencia.

Características del test realizado:

Al igual que para la validación de la hipótesis anterior, para la validación de esta hipótesis se hará uso del total de documentos marcados. En esta ocasión ha sido fijado el número de características a 2000. Para el primer test han sido seleccionadas las 2000 características de frecuencia media y para el segundo se han utilizado aquellas con menor frecuencia.

Resultados obtenidos:

A continuación se muestran las gráficas de F en función del factor C para la prueba realizada. Las gráficas se presentan de izquierda a derecha para máxima frecuencia, frecuencia media y mínima frecuencia respectivamente.



Conclusiones y observaciones:

Observamos que los clasificadores obtenidos para las frecuencias media y mínima no son capaces de clasificar correctamente los documentos. Así confirmamos la hipótesis de que las características de mayor frecuencia tienen también una mayor entropía, es decir, aportan más información sobre el contenido de un documento de opinión.

A la vista de los resultados, en pruebas sucesivas en las que se limite el número de características, únicamente serán seleccionadas aquellas más frecuentes. Aún así, con tan pocos tests realizados no es posible garantizar que dicha partición sea la óptima. Sería muy interesante el desarrollo de un conjunto de pruebas que ayude a descubrir las estrategias óptimas para la selección de características en función de su frecuencia u otro factor para este problema. Sin embargo, este análisis queda fuera del abasto de este proyecto y se deja a investigaciones futuras.

PRUEBA 3 : APLICABILIDAD ENTRE DOMINIOS

Ya ha sido comprobado que es posible encontrar un clasificador binario que sea capaz de discernir entre documentos subjetivos no subjetivos de aplicaciones dentro de un ámbito o ámbitos determinados. Sin embargo aún no sabemos si resulta factible aplicar un modelo construido para un ámbito o ámbitos de aplicaciones determinados en un ámbito diferente.
Parece lógico pensar que siendo la subjetividad un aspecto tan abstracto y general, el ámbito de la aplicación a la que se refieren los comentarios no resulte un factor de excesiva relevancia.

Hipótesis:

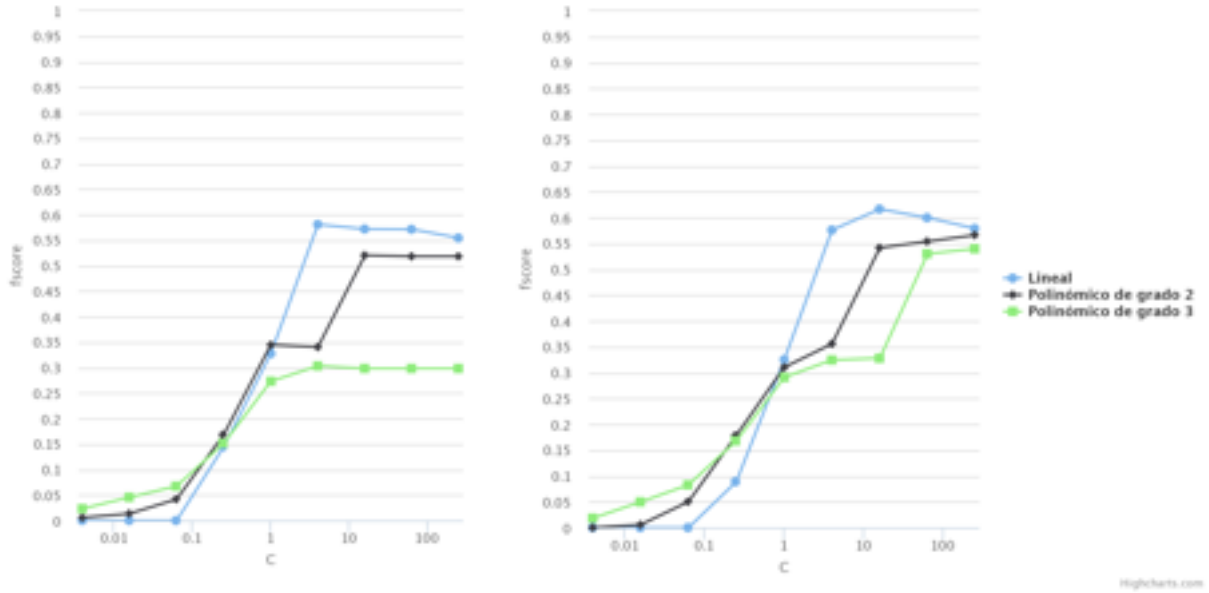
Es posible aplicar un clasificador de comentarios de aplicaciones de un ámbito o ámbitos determinados a comentarios referentes a una aplicación de un ámbito diferente.

Características del test realizado:

Para este test será utilizado un clasificador equivalente al analizado en la primera parte de este experimento. En particular, se utilizarán los comentarios de los ámbitos “mensajería”y “noticias” como corpus de aprendizaje y los comentarios del ámbito “reproductores de música” como corpus de evaluación.No se realizará una reducción en el número de características consideradas, pues queda demostrado en apartados anteriores que es en este caso en el cual el factor de calidad obtiene valores máximos.
Será realizada una prueba adicional utilizando únicamente los ejemplos para el ámbito “mensajería” como corpus de aprendizaje, utilizando el mismo corpus de evaluación. De este modo será posible estudiar la evolución del factor de calidad F conforme aumenta el número de dominios analizados.

Resultados obtenidos:

A continuación se muestran las gráficas de F en función del factor C para la prueba realizada. En la gráfica de la izquierda se muestran los resultados cuando el corpus de aprendizaje se compone únicamente de comentarios del ámbito “mensajería”. En la gráfica de la derecha se muestran los resultados cuando se combinan los ámbitos “mensajería” y “noticias”.



Conclusiones y observaciones:

Vemos que no ha sido posible obtener los valores de calidad mínimos requeridos para ninguno de los dos clasificadores evaluados. Queda por por tanto descartada la hipótesis planteada en este apartado.

Resulta interesante cómo para esta prueba ha sido el kernel lineal el que mejores resultados ha aportado y el de grado 3 el que ha tenido un peor comportamiento. Una posible explicación de este comportamiento es el hecho de que, puesto que el modelo utilizado no es aplicable al ámbito validado, un clasificador que no se adapta tan bien a los documentos de su corpus de entrenamiento resulta más generalista y por tanto ofrece unas mejores resultados en ámbitos diferentes.

También es interesante observar que la mejora en la calidad experimentada al añadir un ámbito más al corpus de entrenamiento ha resultado casi inapreciable. Esto parece indicar que las características que denotan subjetividad y son comunes a los comentarios de una pareja cualquiera de ámbitos son prácticamente los mismos independientemente de la pareja elegida. Una validación más exhaustiva de esta teoría podría resultar muy interesante, sin embargo no disponemos de ejemplos relativos a un número suficiente de ámbitos diferentes para realizar este experimento.

PRUEBA 4 : CASO SEMI-EQUILIBRADO

Para finalizar con el estudio de los clasificadores de documentos subjetivos y no subjetivos, analizaremos una situación hipotética en la cual no existe una diferencia tan elevada en el número de documentos pertenecientes a cada clase, es decir, una situación en la que la distribución de la población de ejemplos es más equilibrada.

Comprobaremos además qué ocurre cuando es aplicado un clasificador para este hipotético escenario a un conjunto de ejemplos del escenario real.

Hipótesis 1:

Si el número de documentos subjetivos y no subjetivos de una muestra poblacional no difieren excesivamente, un clasificador para dicha muestra presenta un factor de calidad mayor que para el caso de una muestra muy desequilibrada

Características del test realizado:

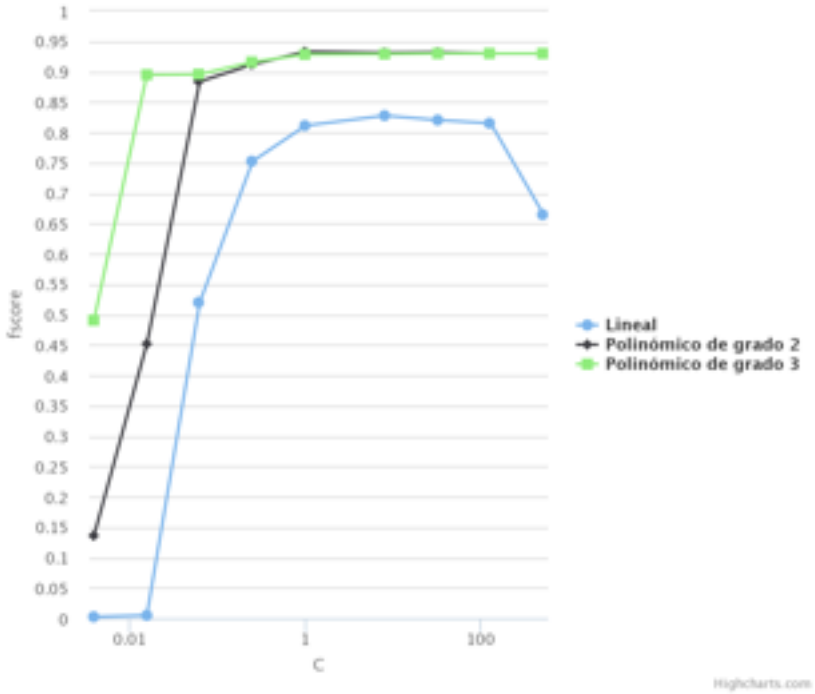
Para este caso se utilizarán ejemplos de los 3 ámbitos de aplicaciones analizados. En particular será seleccionada la totalidad de los documentos pertenicentes a la clase “no subjetivo”. A estos le serán añadidos documentos pertenecientes a la clase “subjetivo” de manera aleatoria, hasta obtener una proporción máxima de 1 - 2. De este modo, la proporción del corpus resultante será de un 33.3% (es decir, tres veces el porcentaje de respecto a la población real) de documentos no subjetivos contra un 66.7% de documentos subjetivos.

Utilizando este método garantizamos que el número de ejemplos de la clase “no subjetivo” se mantenga constante, de modo que no se produzca una merma determinante en la calidad de los clasificadores entrenados debido a una reducción en el número de ejemplos.

Para esta prueba, las características consideradas han sido reducidas siguiendo los criterios óptimos definidos en la segunda parte de este experimento. Es decir, únicamente se han considerado las 2000 características de mayor frecuencia.

Resultados obtenidos:

A continuación se muestran los resultados obtenidos para la prueba realizada:



Conclusiones y observaciones:

En la gráfica anterior se comprueba que el valor del criterio de calidad es considerablemente superior al obtenido en el caso 2, es decir, en el caso de la distribución real. Esto representa un aumento relativo del 24% y de 0.18 en valores absolutos.

Cabe destacar que esto ocurre aún cuando el número de ejemplos en el corpus de entrenamiento para este escenario es considerablemente inferior. Queda por tanto confirmada la hipótesis planteada en este apartado.

De nuevo observamos que la medida del factor de calidad obtenida para el caso del kernel lineal es de más de 0.1 inferior a los valores para los kernels de grado 2 y 3. Vemos por tanto que para este problema no sólo son importantes los tokens que aparecen en las oraciones sino también la relación entre parejas de estos, pues sin una transformación del espacio no es posible dividir linealmente los conjuntos de ejemplos.

Hipótesis 2:

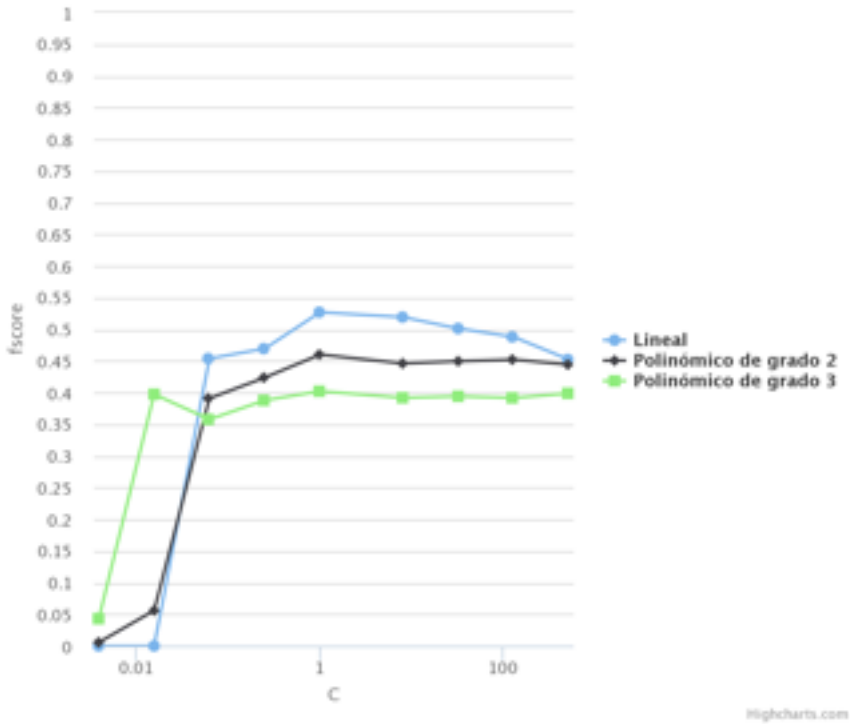
Un clasificador entrenado a partir de un conjunto de ejemplos parcialmente equilibrado no es aplicable a un escenario real.

Características del test realizado:

En esta prueba se aplicará un clasificador para los dominios “mensajería” y “noticias”, previamente equilibrados según la metodología descrita en el apartado anterior, al conjunto de documentos del dominio “reproductores de música”, no equilibrado previamente. Al igual que para la prueba 3, no se realizará una reducción en el número de características considerados.

Resultados obtenidos:

A continuación se muestran los resultados obtenidos para la prueba realizada:



Conclusiones y observaciones:

Vemos que se experimenta una reducción considerable, de 0.08 en valor absoluto o un 13% en valor relativo, del valor de F respecto a los valores obtenidos en las pruebas realizadas durante la parte 3 de este experimento.

Confirmamos de este modo la hipótesis planteada y concluimos por tanto que un clasificador construido a partir de un corpus equilibrado no resulta adecuado para un escenario en el cual la distribución de los documentos resulta muy desequilibrada.

CLASIFICACIÓN SEGÚN LA CARACTERÍSTICA VALORADA

INTRODUCCIÓN

Este experimento tiene por objetivo centrado en analizar clasificadores que atiendan a la polaridad de los documentos, es decir, si son positivos o si por contra son negativos. Este problema puede parecer a priori más complejo que el anterior. Sin embargo veremos que finalmente no es así, y los resultados obtenidos acaban siendo considerablemente mejores que en el caso anterior (evidentemente, según el criterio de calidad definido en esta memoria).
De nuevo utilizaremos exclusivamente clasificadores binarios. Así, las parejas de clases detectadas serán “positivo o no-positivo” y “negativo o no-negativo”. De momento, ignoraremos los documentos neutros, pues serán considerados “no-positivos” y “no-negativos” a la vez.

DISTRIBUCIÓN DE LOS EJEMPLOS

Para el caso de la detección de la polaridad, nos encontramos con un problema en el cual los los corpus se encuentran aproximadamente equilibrados: 7151 contra 6304 en un caso y 4691 contra 8764 en el otro. Así, en el peor caso tendremos una población mínima del 35% y en ningún caso habrá más del doble de ejemplos de una clase que de la otra.

CONSIDERACIONES ADICIONALES

Puesto que ya disponemos de una población equilibrada, no será necesario estudiar escenarios alternativos. De nuevo, compararemos los resultados obtenidos utilizando un clasificador dentro del mismo dominio de los ejemplos con los que fue construido respecto a un clasificador distinguiendo ejemplos de un dominio diferente.
Para cada uno de los tests, los resultados obtenidos para las polaridades positivas y negativas serán presentadas conjuntamente y se comentarán las diferencias que encontremos entre ambos.

PRUEBA 1: CASO GENERAL

Al igual que se realizó para el experimento anterior, determinaremos inicialmente si es posible construir un clasificador SVM que sea capaz de dividir aquellos problemas que contienen valoraciones positivas de aquellos que no, es decir, de aquellos que contienen valoraciones negativas, neutras o no contienen ninguna valoración. El mismo procedimiento será realizado para el caso de las valoraciones negativas
Las conclusiones obtenidas estarán de nuevo condicionadas a que el clasificador utilizado haya sido entrenado mediante ejemplos pertenecientes al mismo ámbito o ámbitos de aplicación de los comentarios que se desean clasificar.

Hipótesis:

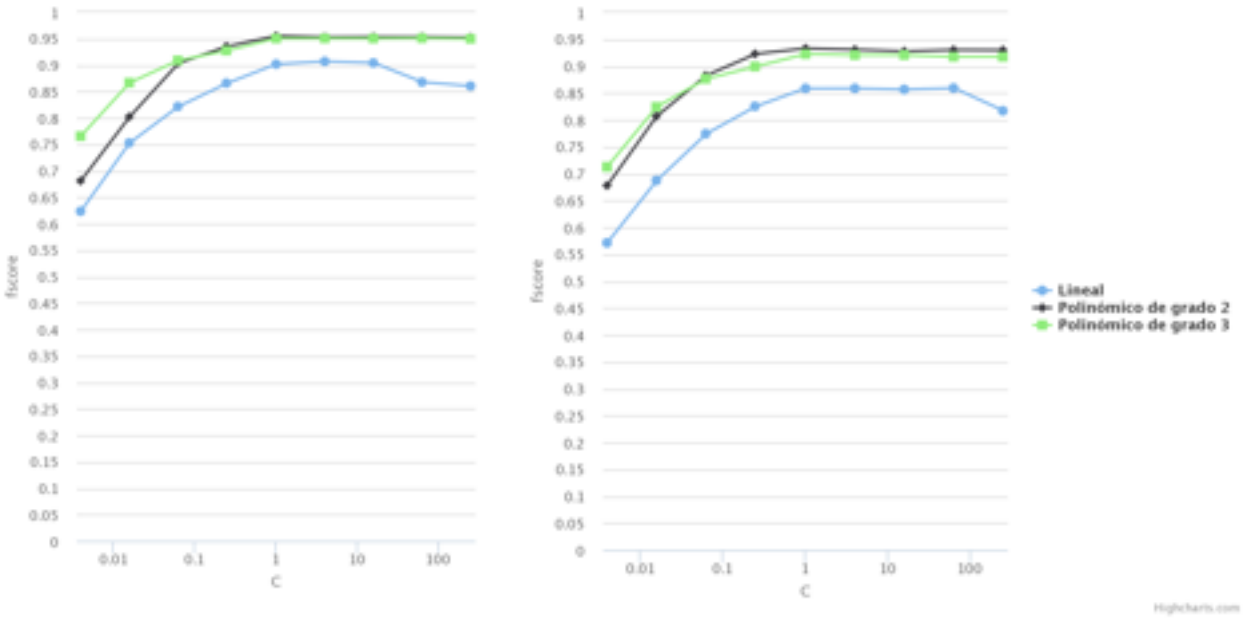
Si no se realiza reducción en el número de características evaluadas, es posible encontrar un clasificador basado en SVM que sea capaz de separar documentos positivos de aquellos que no lo son con un factor de calidad que supere el valor de referencia mínimo planteado en este proyecto.

Características del test realizado:

Las metodología seguida para este test es análoga a la realizada durante la primera parte del experimento 1, con el fin de poder comparar los resultados obtenidos.
De nuevo, únicamente serán utilizados los comentarios correspondientes a dos de los tres dominios de aplicaciones estudiados y no se realizará reducción alguna al número de características consideradas por el clasificador.

Resultados obtenidos:

A continuación se muestran los resultados obtenidos para la prueba realizada. A la izquierda se muestran los valores de F en función de C obtenidos para el clasificador de polaridad “positiva” y “no-positiva”. A la derecha se muestra la gráfica análoga para el clasificador de polaridad “negativa” y “no-negativa”:



Conclusiones y observaciones:

A la vista de los resultados queda comprobada la veracidad de la hipótesis planteada en esta prueba. Confirmamos por tanto que es posible conseguir valores de F superiores a los los límites inferiores definidos para este proyecto para cualquiera de los kernels utilizados.

Es de especial interés la comparación de estos resultados con los obtenidos en la primera prueba del experimento anterior. Tanto para el caso de la polaridad positiva como para la polaridad negativa, los valores pico de F obtenidos resultan considerablemente superiores a los obtenidos para el problema la clasificación de documentos subjetivos y no subjetivos.

Esto resulta lógico si consideramos el hecho de que para este problema la distribución de los ejemplos es mucho más equilibrada. Como ya concluimos en la prueba 4 del experimento anterior, un problema en el que los corpus están equilibrados resulta más sencillo que un problema no equilibrado según el criterio de calidad elegido para este proyecto.

Otro aspecto de interés es el comportamiento del valor de F en función de C para el caso del kernel lineal. Vemos que comienza siendo creciente, llega a un cierto valor de pico y después decrece. Este comportamiento parece ser debido al efecto de sobreajuste u *overfitting*, detallado en apartados anteriores de esta memoria.

PRUEBA 2 : REDUCCIÓN DEL NÚMERO DE CARACTERÍSTICAS

En esta segunda parte del experimento comprobaremos si las conclusiones tomadas durante el primer experimento sobre la relevancia de la frecuencia de las características son también aplicables al problema de la clasificación según polaridad.

En esta ocasión no se realizará un análisis exhaustivo del problema sino que únicamente se valorará el impacto en la calidad de un clasificador de polaridad cuando se utiliza el criterio de reducción del número de atributos presentado en el mencionado experimento.

Hipótesis:

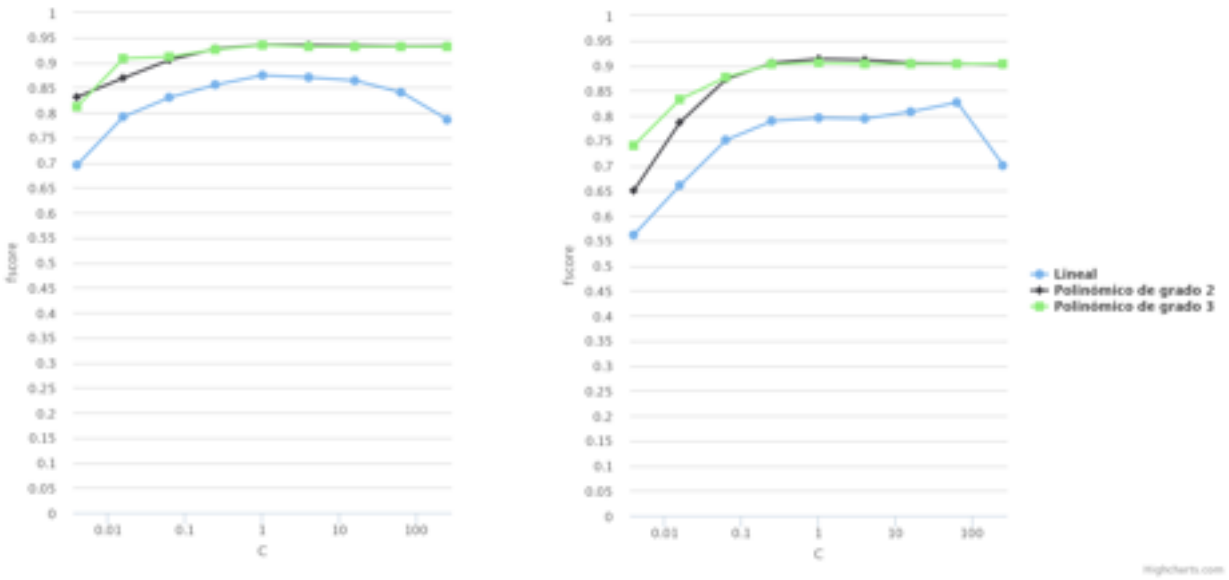
Es posible construir un clasificador binario de documentos según su polaridad utilizando únicamente una parte de características consideradas, sin que esto repercuta en una merma considerable de la calidad respecto a un clasificador para el cual no se realiza tal reducción.

Características del test realizado:

Para la realización de esta prueba ha sido utilizado el total de documentos marcados. La estrategia de selección de características ha sido la misma que seguida en el experimento anterior. Es es decir, limitando el número de características a las 2000 de mayor frecuencia de aparición en el conjunto de ejemplos.

Resultados obtenidos:

A izquierda y derecha se encuentran los resultados para el clasificador de polaridad positiva y el de polaridad negativa respectivamente:



Conclusiones y observaciones:

Comprobamos que la hipótesis planteada también se confirma para el problema de la clasificación según la polaridad de los documentos. Además se puede apreciar que la reducción en la calidad de los clasificadores obtenidos resulta prácticamente inapreciable respecto al caso sin reducción (Inferior a 0.02 para la el clasificador de polaridad positiva, representado una caída de apenas un 2.1%).

Confirmamos de esta manera que para la problemática planteada en este proyecto, resulta más sencillo un clasificador de comentarios según la polaridad que uno de comentarios subjetivos y no subjetivos. Además vemos que la limitación del número de características según su frecuencia no solo es también aplicable al caso de la clasificación de la polaridad sino que su repercusión negativa en la calidad del clasificador obtenido es aún menor.
Observamos de nuevo grandes similitudes entre los resultados obtenidos para las polaridades positiva y negativa. Confirmamos por tanto que se trata de problemas de similar dificultad.

Esto puede ser debido a que, puesto que el número de documentos no subjetivos o neutros es prácticamente despreciable, un clasificador de documentos positivos o “no-positivos” es aproximadamente equivalente a un clasificador de documentos positivos y negativos.

PRUEBA 3 : APLICABILIDAD ENTRE DOMINIOS

Finalizaremos este experimento estudiando si es posible aplicar un modelo de clasificación según subjetividad a los comentarios de un dominio de aplicaciones a partir del cual no fue entrenado. Ya hemos visto que la calidad de los resultados obtenidos para el caso de la clasificación de documentos subjetivos y no subjetivos no llega a los mínimos exigidos en este proyecto. A continuación comprobaremos si esta limitación se mantiene para el problema de la clasificación según polaridad.

Hipótesis:

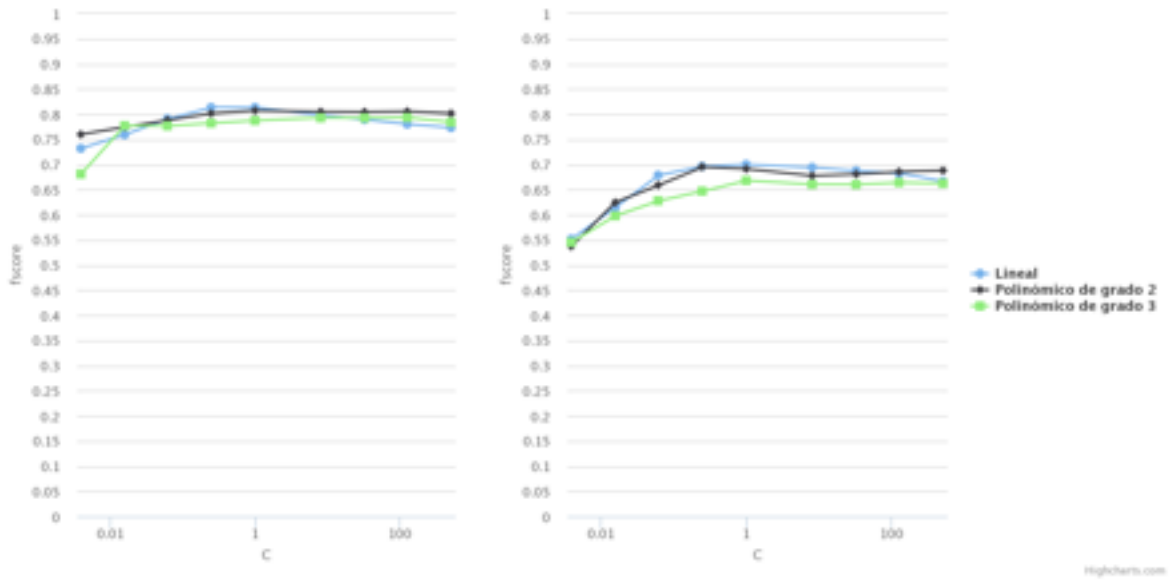
Es posible aplicar un clasificador de documentos según su polaridad entrenado a partir de comentarios de aplicaciones un ámbito o ámbitos determinados a comentarios referentes a una aplicación de un ámbito diferente.

Características del test realizado:

Los parámetros de este test son análogos a los del test realizado para el experimento 1. Los comentarios de los ámbitos “mensajería”y “noticias” han sido utilizados como corpus de aprendizaje y los comentarios del ámbito “reproductores de música” como corpus de evaluación. No ha sido realizada tampoco una reducción en el número de características consideradas.

Resultados obtenidos:

A izquierda y derecha se muestran los resultados para el clasificador de polaridad positiva y el de polaridad negativa respectivamente:



Conclusiones y observaciones:

A diferencia de lo ocurrido en el experimento 1, en esta ocasión podemos observar que que el valor de F supera el umbral mínimo marcado y por tanto podemos confirmar la hipótesis planteada.

Resulta interesante observar la considerable diferencia existente entre los valores máximos obtenidos para el clasificador de documentos positivos y el clasificador de documentos negativos, de más de 0.10 en el valor de F.

De este resultado podemos suponer que existe un mayor grado de similitud entre los documentos positivos para ámbitos de aplicaciones diferentes que para los documentos negativos, es decir, la clasificación de documentos con valoraciones positivas resulta más generalizable.

CLASIFICACIÓN MULTI-CLASE SEGÚN LA POLARIDAD

INTRODUCCIÓN

En este experimento continuaremos con el estudio de los clasificadores de comentarios según la polaridad. En esta ocasión valoraremos las diferencias que se pueden apreciar cuando se utiliza un clasificador multiclase.
Como se detalla en apartados anteriores de la memoria, utilizaremos dos tipos diferentes de clasificadores multiclase: uno especialmente diseñado para el problema que queremos tratar, basado en la unión de dos clasificadores binarios; y otro haciendo uso de la implementación multiclase provista por la librería LibSVM.

DISTRIBUCIÓN DE LOS EJEMPLOS

Puesto que en este experimento también se estudia la clasificación según la polaridad, la distribución de los ejemplos es la misma que para el experimento anterior. Sin embargo cabe recordar que en este caso se introduce un valor de polaridad adicional: la polaridad neutra.
La distribución final es por tanto de 7151 documentos con polaridad positiva, 4691 documentos con polaridad negativa, 124 con polaridad neutra y 1489 documentos no subjetivos.

CONSIDERACIONES ADICIONALES

El clasificador multiclase específico para esta plataforma se basa en el cálculo de la intersección de los conjuntos “polaridad positiva o neutra” y polaridad negativa o neutra”. Por tanto, el primer paso a realizar es la construcción de dos clasificadores binarios, uno que sea capaz de detectar documentos con polaridad positiva o neutra y un segundo que detecte documentos con polaridad negativa o neutra.
En este experimento nos centraremos únicamente en comparara los resultados obtenidos mediante la estrategia de clasificación multi-clase presentada en esta memoria. No se trata por tanto de un estudio exhaustivo sobre la aplicación de clasificadores multi-clase al ámbito de la clasificación de comentarios de aplicaciones móviles. Un estudio de estas características resultaría mucho más extenso, es por esto que queda fuera del abasto de este proyecto.

PRUEBA ÚNICA : ALTERNATIVAS DE LA CLASIFICACIÓN MUTI-CLASE

A continuación estudiaremos la aplicación de la estrategia de clasificación multi-clase específica para este proyecto, presentada en apartados anteriores de esta memoria. Además compararemos los resultados obtenidos con los correspondientes a un clasificador multi-clase genérico.
Los factores de calidad que utilizaremos en esta prueba serán la exactitud (accuracy) y la F1 media, pues se adecua más a los requerimientos de este problema particular. En la sección correspondiente de apartado de metodología es posible encontrar una justificación más extensa de esta decisión.

Hipótesis:

Un clasificador multi-clase basado en la estrategia planteada en esta memoria presenta un factor de calidad superior a un clasificador multi-clase genérico para el caso tratado.

Características de la prueba realizada:

Puesto que ya se ha confirmado que resulta viable la aplicación de un clasificador de polaridad entre diferentes dominios y debido a las limitaciones de tiempo de las que se disponía, únicamente serán comprobados los resultados para este caso extremo.

Los corpus de entrenamiento y evaluación serán por tanto iguales a los utilizados en el paso 3 de experimento 2. Es decir, aplicando un modelo construido a partir de ejemplos de los ámbitos “mensajería” y “noticias” al ámbito “reproductores multimedia”, sin reducir el número de dimensiones.

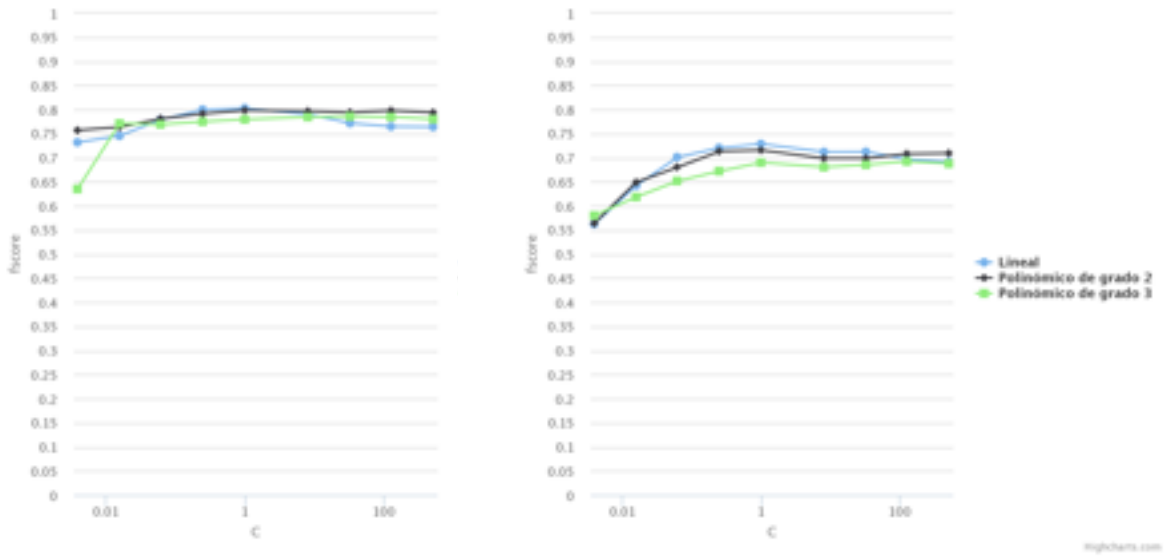
No ha sido considerado necesario realizar la prueba para el caso genera, pues el fin de este experimento es meramente comparativo y los resultados pueden ser fácilmente extrapolables a dicho caso.

PASOS PREVIOS

Como paso previo a la evaluación de la estrategia de clasificación multi-clase planteada en este proyecto, evaluaremos por separado los clasificadores de los que se compone: el clasificador binario de polaridad positiva o neutra y el clasificador binario de polaridades negativa o neutra.
Es de esperar que los resultados obtenidos para los clasificadores de polaridad “positiva o neutra” y “negativa o neutra” no difieran en exceso de los resultados obtenidos para polaridades “positiva” y “negativa”.

Resultados obtenidos (Paso Previo):

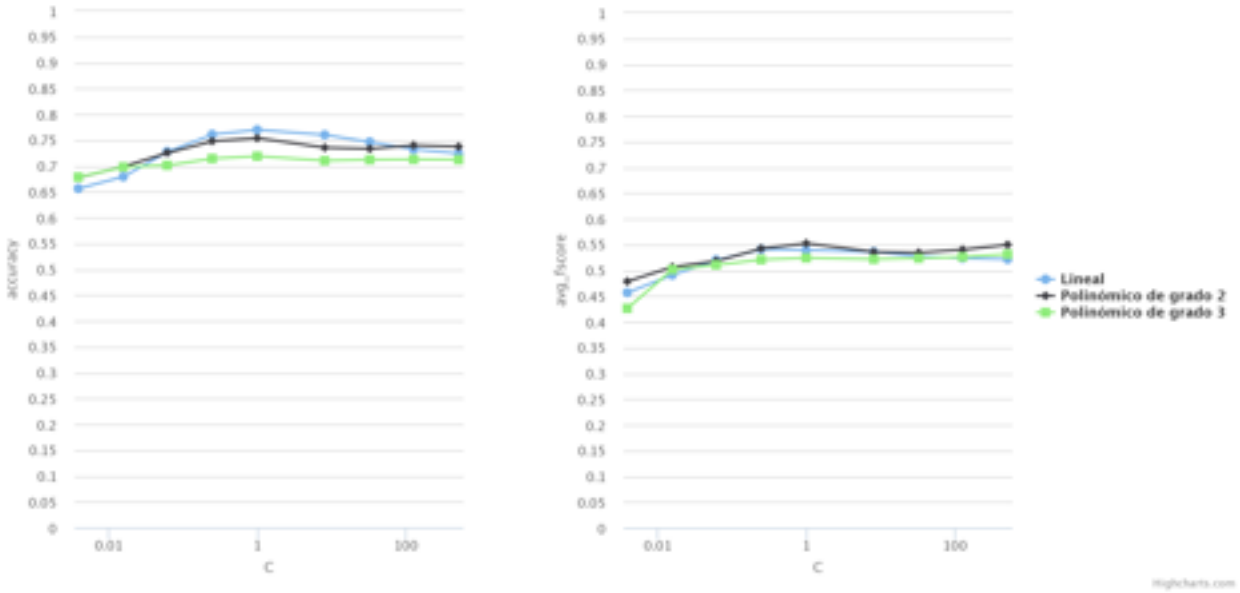
A izquierda y derecha se muestran los resultados para el clasificador de polaridad positiva o neutra y el de polaridad negativa o neutra respectivamente:



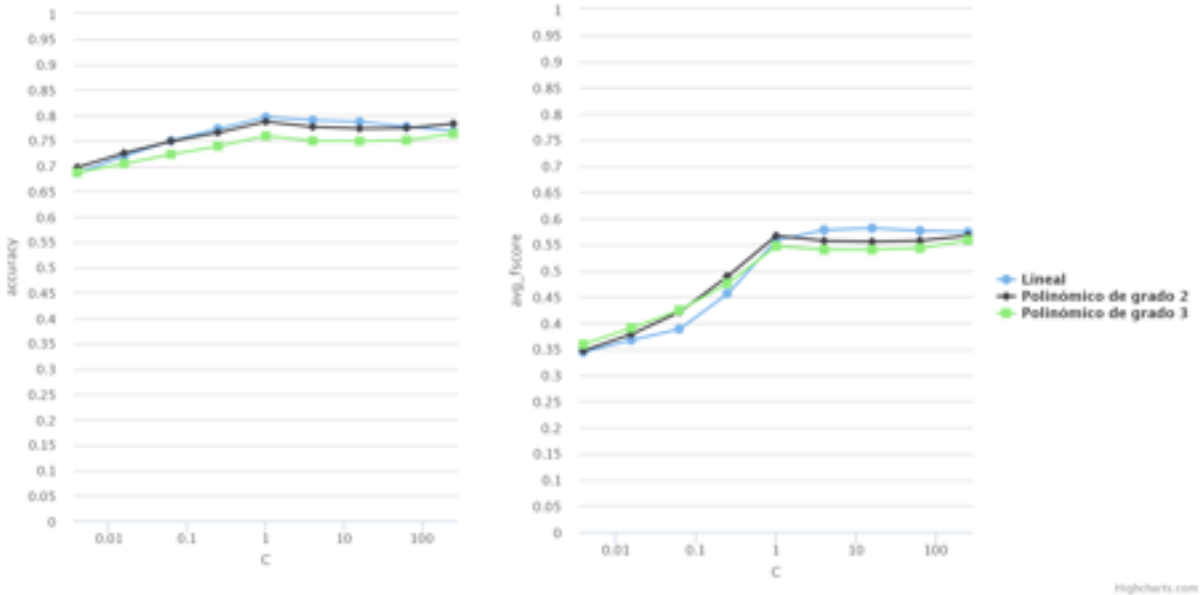
Vemos que efectivamente los resultados obtenidos para los clasificadores de polaridad positiva o neutra y negativa o neutra son cercanos a los factores de calidad conseguidos para los clasificadores de polaridad positiva y negativa y que por tanto la adición de la polaridad neutra no afecta visiblemente a la calidad de los mismos.

Resultados obtenidos:

A continuación se muestran los resultados obtenidos para la combinación de los resultados obtenidos por los clasificadores anteriormente evaluados. A izquierda y derecha se muestran los valores de accuracy y F media en función de C respectivamente:



Los resultados obtenidos para el clasificador muti-clase de LibSVM son los siguentes:



Conclusiones y observaciones:

Comprobamos que no existen grandes diferencias en los valores de los factores de calidad para uno y otro algoritmo de clasificación multi-clase. Sin embaro, el algoritmo de clasificación mutilase proporcionado por LibSVM obtiene unos resultados ligeramente superiores para el factor *accuracy;* siendo este aproximadamente 0.035 superior ,es decir, una mejora del 4.37%. Bajo esta justificación, rechazamos por tanto la hipótesis planteada en este experimento.

Concluimos por consiguiente que el uso de un clasificador multi-clase multipropósito resulta más adecuado para este problema. Además existe un factor adicional para preferir esta opción respecto de la anteriormente mencionada: El tiempo de procesado para el clasificador multipropósito es de aproximadamente la mitad respecto al algoritmo por combinación, en particular el tiempo medio de cálculo de entrenamiento de cada clasificador fue de 19.2 minutos y 42.4 minutos respectivamente.

A la vista de los resultados, podemos plantear dos factores que han podido desencadenar un peor funcionamiento del clasificador por combinación respecto al general:

- El error asociado a cada uno de los clasificadores planteados se combina siempre de manera aditiva. Es decir, no existe ningún caso en el que si ambos clasificadores ofrecen una respuesta errónea el resultado se correcto y siempre que se produce un resultado erróneo en alguno de ellos, la evaluación final resultará también errónea.
- La optimización de los parámetros de cada clasificador binario se realiza de forma totalmente independiente a la calidad del clasificador final.

CLASIFICACIÓN SEGÚN LA CARACTERÍSTICA VALORADA

INTRODUCCIÓN:

Para este último experimento, construiremos clasificadores binarios capaces de identificar la característica o tema valorada en un comentario de entre las 5 características presentadas en los anteriores apartados de esta memoria.
Estudiaremos también la posibilidad de clasificar los documentos según la polaridad restringida a una cierta característica, es decir, si se realiza una valoración positiva o negativa de la característica comentada.

DISTRIBUCIÓN DE LOS EJEMPLOS:

De nuevo nos encontramos con un caso en el cual la distribución de la población de ejemplos no se encuentra equilibrada. Además, las proporciones varían considerablemente para cada uno de los temas. A continuación se detalla la distribución poblacional para cada una de las características de una aplicación tratados en este proyecto:

- Facilidad de Uso: 391 de los 13455 ejemplos (lo que supone un 2.9% del total).
- Rendimiento: 758 de los 13455 ejemplos (lo que supone un 5.63% del total).
- Estabilidad: 1119 de los 13455 ejemplos (suponiendo un 8.32% del total).
- Diseño: 622 de los 13455 ejemplos (lo que equivale un 4,62% del total).
- Utilidad: 1922 de los 13455 ejemplos (o lo que es igual, un 14.28% del total).

CONSIDERACIONES ADICIONALES

Pese a que los resultados obtenidos para cada categoría se presenten consecutivos, el estudio de los resultados para cada una de las mismas se realizará independientemente del resto.
No es por tanto una buena práctica realizar extrapolaciones de los resultados obtenidos en una categoría a el resto, pues las características de los documentos pertenecientes a cada una de ellas no tienen por que coincidir; especialmente si tenemos en cuenta las considerables diferencias que encontramos en la proporción entre los ejemplos positivos y negativos.

PRUEBA 1 : REDUCCIÓN DEL NÚMERO DE CARACTERÍSTICAS

Al igual que para los experimentos 1 y 2, comenzaremos por estudiar si es posible construir un clasificador de documentos atendiendo a el tema o característica valorada.

Para este experimento se ha obviado el caso general, es decir, sin reducción del número de características. De confirmarse la hipótesis planteada para el caso con reducción del número de características, también quedará confirmado el caso general.

Las conclusiones obtenidas estarán de nuevo condicionadas a que el clasificador utilizado haya sido entrenado mediante ejemplos pertenecientes al mismo ámbito o ámbitos de aplicación de los comentarios que se desean clasificar.

Hipótesis:

Es posible encontrar un clasificador basado en SVM que sea capaz de separar documentos positivos de aquellos que no lo son con un factor de calidad que supere el valor de referencia mínimo planteado en este proyecto.

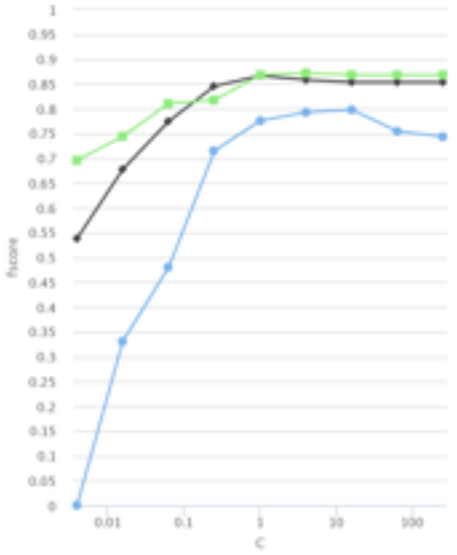
Características de la prueba realizada:

Para esta evaluación se considerará el total de documentos marcados, y será reducido el número de características a las 2000 más comunes; al igual que se hizo para las la parte 2 de los experimentos 1 y 2.

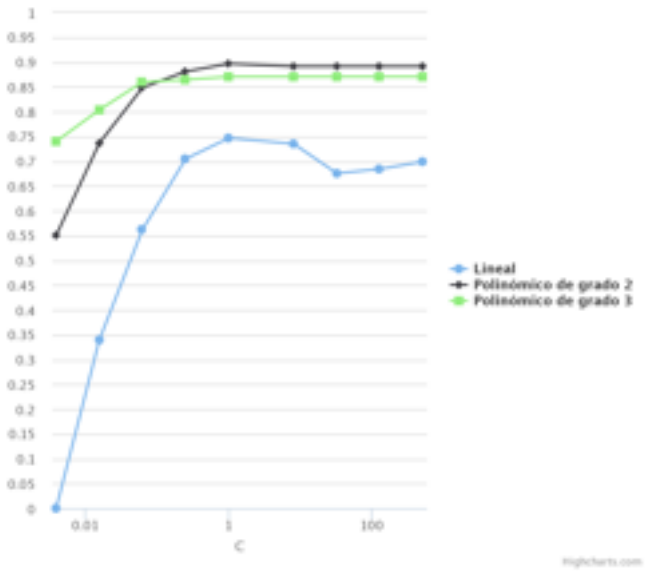
Resultados obtenidos:

A continuación se listan los resultados obtenidos para cada tema clasificado:

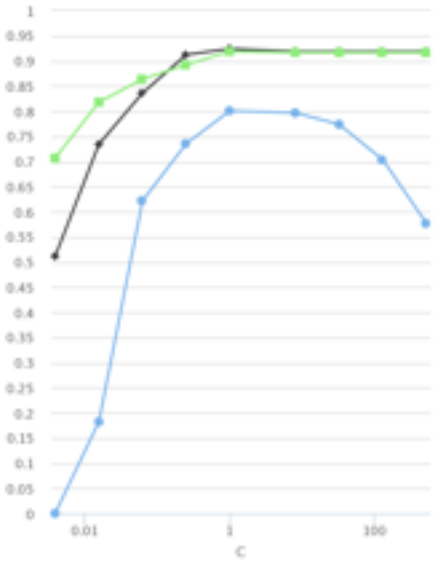
Diseño:



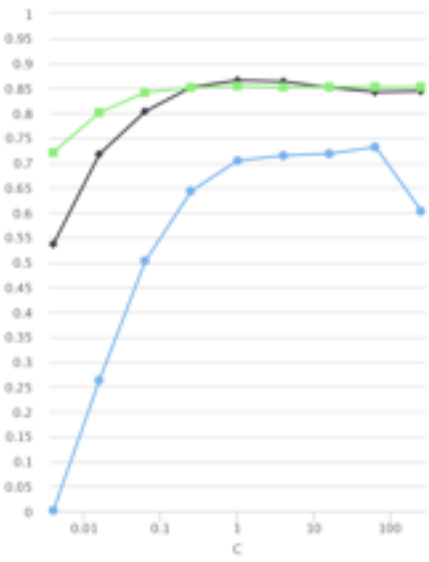
Facilidad de uso:



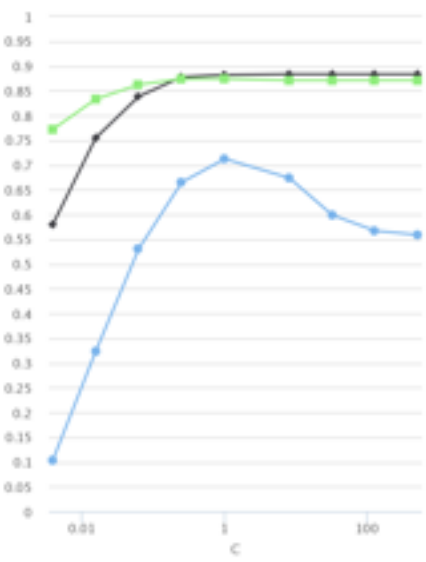
Estabilidad:



Rendimiento:



Utilidad:



Conclusiones y observaciones:

En la totalidad de los casos observados el valor de F supera el umbral mínimo marcado. Por tanto podemos confirmar la hipótesis planteada para todas las características valoradas estudiadas en este proyecto.

Observando las gráficas de resultados para cada una de las características vemos que, pese a que a priori parezca un problema de mayor complejidad, la respuesta del clasificador presenta unos valores de calidad superiores que para los problemas de clasificación de subjetividad y cercanos o iguales a los de clasificación de polaridad.

Podemos explicar este comportamiento atendiendo a la mayor concreción de este problema. Si analizamos los ejemplos de cada uno de los casos planteados, observamos que existen determinadas palabras recurrentes a todos los los comentarios referentes a una misma categoría. Además, podemos observar que para una una aplicación en concreto, los comentarios relativos a una característica determinada suelen ser recurrentes y es posible encontrar diversos comentarios que valoran el mismo aspecto.

El hecho de que exista una correlación tan grande entre los comentarios de una determinada característica conlleva que los ejemplos del corpus de aprendizaje y test no difieran en exceso y por tanto los resultados obtenidos por este tipo de clasificadores resulten muy favorables.

PRUEBA 2 : APLICABILIDAD ENTRE DOMINIOS

Siguiendo con la metodología aplicada al resto de experimentos, evaluaremos ahora la posibilidad de aplicar un clasificador de documentos para dos dominios determinados a un dominio diferente.

Hipótesis:

Es posible aplicar un clasificador de la característica valorada en un comentario de un ámbito o ámbitos de aplicaciones determinados a comentarios referentes a una aplicación de un ámbito diferente.

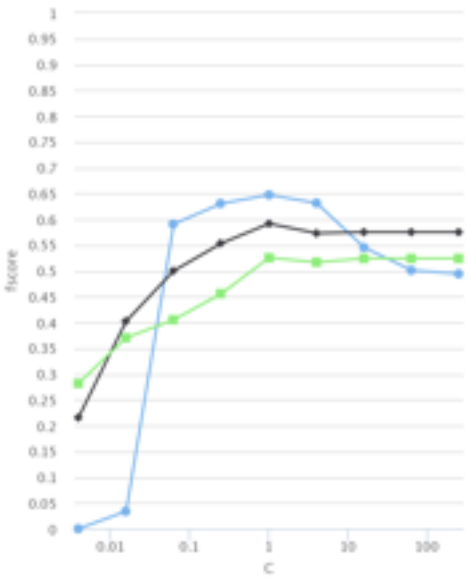
Características del test realizado:

Los comentarios de los ámbitos “mensajería”y “noticias” han sido utilizados como corpus de aprendizaje y los comentarios del ámbito “reproductores de música” como corpus de evaluación. No ha sido realizada una reducción en el número de características consideradas.

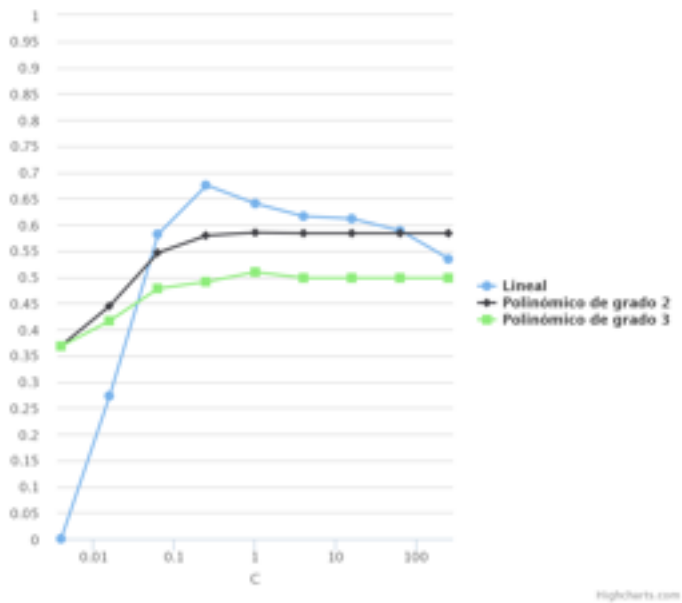
Resultados obtenidos:

A continuación se listan los resultados obtenidos para cada tema clasificado:

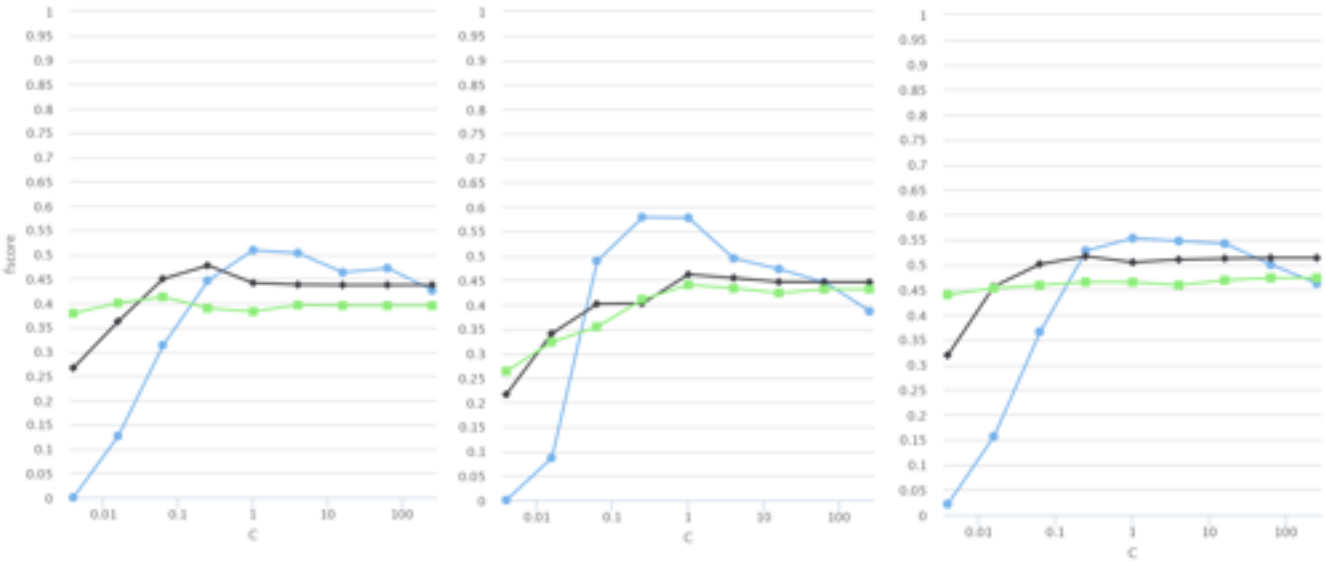
Diseño:



Facilidad de uso:



Estabilidad: Rendimiento: Utilidad:



Conclusiones y observaciones:

Para este caso, al igual que para la clasificación de documentos subjetivos y no subjetivos, los valores obtenidos de F no superan el umbral marcado y por tanto podemos considerar la hipótesis rechazada.

Este resultado resulta coherente con las conclusiones realizadas en la primera parte de este experimento sobre la concreción de los comentarios tratados. Vemos por tanto que puesto que existe una mayor correlación entre los comentarios que valoran alguna característica para una aplicación o ámbito de aplicación determinado, un clasificador especializado en dicho dominio no es generalizable a otras aplicaciones o ámbitos.

De nuevo observamos que para este tipo de problemas, la respuesta del clasificador de kernel lineal resulta mejor que para aquellos con kernel de un grado superior. Esto nos permite reafirmarnos en la suposición enunciada en el experimento 1.

PRUEBA 3 : POLARIDAD EN UNA CARACTERÍSTICA

Intentando ser aún más precisos en la clasificación de los comentarios, estudiaremos ahora la viabilidad de diseñar clasificadores que detecten la polaridad de los comentarios respecto a una característica determinada.
Cabe aclarar que en esta prueba únicamente se presentarán los resultados obtenidos para la clasificación de documentos con polaridad positiva pues como ya observamos en pruebas anteriores.

Hipótesis:

Es posible encontrar un clasificador de comentarios que sea capaz de de clasificar documentos atendiendo a su polaridad respecto de una característica determinada con un factor de calidad que supere el valor de referencia mínimo planteado en este proyecto.

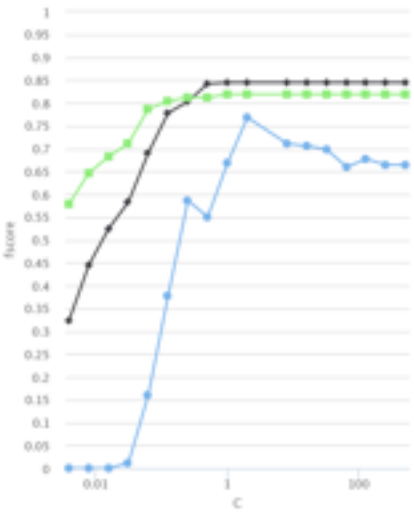
Características del test realizado:

Para este test han sido utilizados los mismos parámetros considerados en la primera parte de este experimento.

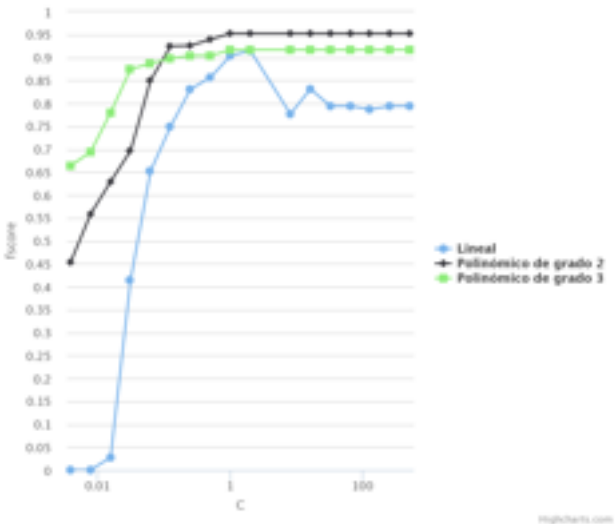
Resultados obtenidos:

A continuación se listan los resultados obtenidos para cada tema clasificado:

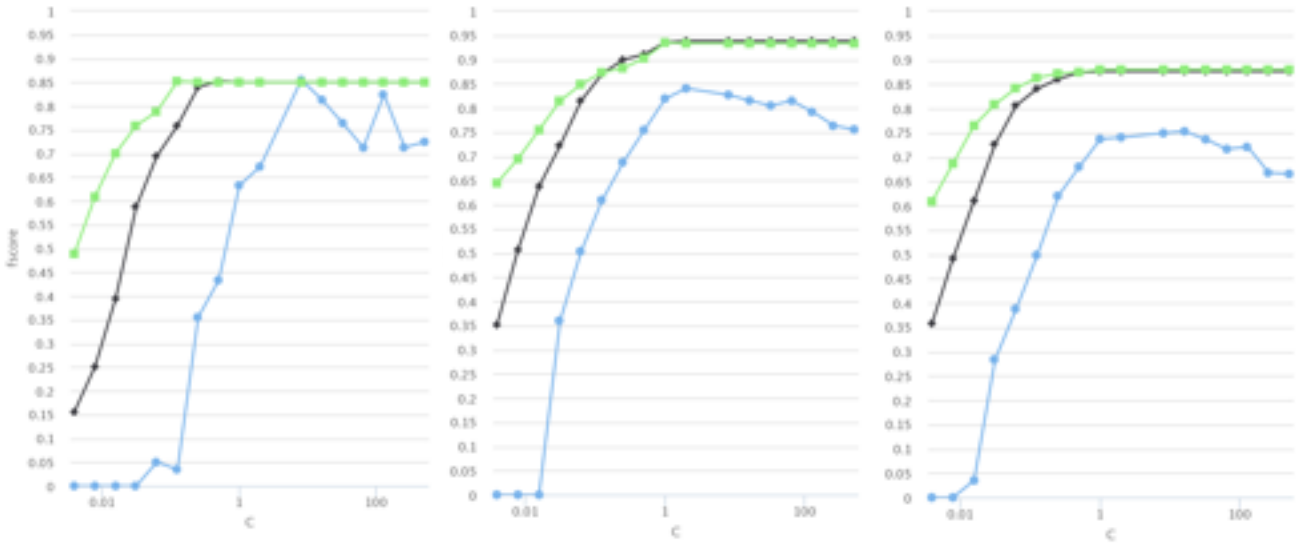
Diseño:



Facilidad de uso:



Estabilidad: Rendimiento: Utilidad:



Conclusiones y observaciones:

En la totalidad de los casos observados el valor de F supera el umbral mínimo marcado. Por tanto podemos confirmar la hipótesis planteada para todas las características valoradas estudiadas en este proyecto.

En contra de lo que podría esperarse, incluso para el caso de la detección de la polaridad restringida a una característica particular los resultados obtenidos son francamente favorables. Obtenemos por tanto valores de F mínima cercanos a 0.9 cuando el valor de C es superior a 1 para todas las características valoradas. Volvemos a comprobar además que para este problema el grado óptimo para un clasificador basado en SVM de kernel polinómico es 2, pues utilizando un kernel de grado superior únicamente obtenemos una mejor en los resultados para valores muy reducidos de C.

Otra conclusión que podemos obtener de estos resultados es que el uso de un SVM de kernel lineal no resulta útil para la resolución de este problema, pues sus valores medios quedan considerablemente alejados de los obtenidos para kernels de grados superiores.

Atendiendo a la forma de las gráficas obtenidas, no parece apreciarse un efecto de overfitting para ninguno de los casos. Esto puede deberse a que, una vez realizada la transformación por medio de la función de kernel, los conjuntos de ejemplos positivos y negativos son fácilmente separables y por tanto no obtenemos grandes diferencias en el hiperplano separador que se obtiene para cada uno de los posibles valores de C.

CONCLUSIONES

INTRODUCCIÓN

En este apartado realizaremos una valoración global de los resultados obtenidos y estudiaremos la viabilidad de la construcción de un hipotético sistema que haga uso de los modelos desarrollados durante el desarrollo de este proyecto.
Comentaremos además los aspectos en los que ha sido necesario reducir el abasto de este proyecto para adecuarlo a las limitaciones de tiempo y recursos de los que disponíamos.
Continuaremos detallando las maneras en que podría ser extendido este proyecto así como otros escenarios alternativos que valdría la pena considerados con el fin de profundizar aún más en la materia estudiada.

CONCLUSIONES

Atendiendo a los resultados obtenidos para cada una de las pruebas, podemos confirmar que la aplicación de clasificadores basados en SVM resultan perfectamente viable para identificar tanto el tema como la polaridad de comentarios para diversos tipos de aplicaciones móviles. La efectividad está sin embargo limitada por los siguientes condicionantes:

- El ámbito o ámbitos de las aplicaciones sobre las que se valora en los comentarios clasificados ha de ser un subconjunto de los ámbitos de los ejemplos utilizados para la construcción de dichos clasificadores. La merma en la calidad de las soluciones obtenidas cuando se utiliza un clasificador para ámbitos diferentes de los analizados es muy considerable respecto a la calidad de clasificadores dentro del mismo ámbito.
- Reducir el número de tokens considerados al generar un clasificador repercute inevitablemente en una merma de la calidad de las soluciones ofrecidas por dicho clasificador. Este efecto es especialmente considerable cuando los tokens utilizados son de baja frecuencia.
- La clasificación de conceptos más generales como el hecho de que un documento sea o no subjetivo resulta más compleja que para el caso de conceptos más concretos como los documentos que valoran positivamente el diseño. Esto puede deberse entre otras cosas a la mayor ambigüedad de los ejemplos para el primer caso.

Finalmente, si atendemos a las características de los SVM óptimos para cada uno de los casos tratados, llegamos a las siguientes conclusiones:

- El coste computacional asociado a la construcción de los clasificadores aumenta considerablemente para casos equilibrados, es decir, cuando existe aproximadamente el mismo número de documentos de cada clase. Este comportamiento resulta lógico si atendemos al número de parejas de puntos que han de ser comparados cuando se realiza el cálculo del hiperplano divisor.
- El uso de SVM con kernel lineal no resulta adecuado para la práctica totalidad de escenarios planteados. Sin embargo se puede apreciar que para los casos en los que se aplicaban modelos para unos dominios en un dominio diferente, los valores de calidad obtenidos para este kernel igualan o superan a los obtenidos. En este tipo de escenarios, la mayor simplicidad y permisividad de los clasificadores con kernel lineal conlleva que se produzca un menor overfitting y que, pese a que no se obtienen valores óptimos en la aplicación del clasificador sobre el corpus de entrenamiento, para el corpus de test los valores de calidad resultan ser superiores.
- En general resulta preferible el uso de SVM de kernel polinómico de grado 2. Los valores de calidad obtenidos para los kernels de grado 2 y grado 3 respectivamente no difieren en gran medida y en muchos casos son mejores para el primer kernel que para el segundo. De este modo vemos que el considerable aumento en el tiempo de generación de los SVM con kernel polinómico de grado 3 no se traduce en mejores prestaciones y concluimos que el kernel óptimo en el caso estudiado en este proyecto es el de grado 2.
- Pese a que su valor óptimo varía, los valores de calidad obtenidos para valores de C elevados resultan considerablemente superiores. Además, pese a que en algunas de las pruebas resulta apreciable, el efecto de overfitting es muy reducido incluso para valores muy elevados de C.

EXTENSIONES POSIBLES

Como ya se detalla en la introducción, gran cantidad de las pruebas que que se previeron durante la concepción de este proyecto fueron imposibles de realizar debido a las limitaciones en recursos y tiempo de los que disponíamos.
Existen gran cantidad de aspectos en los que este proyecto podría ser extendido, ya sea dentro del ámbito de la investigación en forma de publicaciones científicas o mediante una aplicación práctica de todo el conocimiento adquirido gracias al mismo.
A continuación se listan algunas de las extensiones que consideramos más importantes o con las que podríamos aportar aún más valor al estudio realizado en este proyecto:

- Aumentar el número de dominios analizados con el fin de ser capaces de extrapolar los resultados obtenidos a un caso más general. Debido a que la construcción de los corpus es larga y tediosa, únicamente fue posible el marcado de ejemplos de los ámbitos “reproductores de música”, “aplicaciones de noticias” y “aplicaciones de mensajería”. Para poder generalizar las conclusiones que hemos obtenido a un caso genérico, sería necesario realizar pruebas con diferentes un número superior de dominios.
- Comparar los resultados obtenidos con los que se obtendrían si se hubiesen utilizado SVM con las funciones de kernel RBF o Sigmoidea. También podría resultar de interés optimizar para cada caso los parámetros propios de kernels polinómicos inhomogéneos y r.
- Comparar los resultados con los que obtendríamos con otros algoritmos de clasificación no basados en cálculos vectoriales como AdaBoost.
- Comprobar otras estrategias de selección de tokens para los experimentos con un número de dimensiones limitado. Para esta memoria solo se han contemplado 3 estrategias de selección de tokens diferentes, todas basadas en la frecuencia de los mismos en el total de ejemplos. Sería por tanto de interés realizar un estudio más exhaustivo de otras estrategias de selección de tokens como podrían ser el borrado de tokens que no aportan información basado en diccionario o la aplicación de un corrector o un diccionario de sinónimos.
- Estudiar cómo es posible aprovechar la información adicional de los distintos tokens que obtenemos a partir de la librería de análisis de textos Freeling. Para el caso estudiado en esta memoria, sólo se han utilizado las herramientas de Freeling para la tokenización de los textos y la identificación de las oraciones contenidas en los mismos. Freeling proporciona también otra información que podría ser relevante como la categoría gramatical, el lema o incluso las correferencias dentro del texto.

- Añadir información adicional a las oraciones clasificadas, como pueden ser la posición dentro del comentario (es decir, si se trata del título o el contenido del mismo) u otra información relativa al contexto como los diferentes tokens que contienen las oraciones adyacentes a la misma dentro del texto. Otra opción a considerar podría ser la adición de información relativa a la aplicación a la cual están referidas, como puede ser el nombre, la valoración global, la categoría, etcétera.

OTROS ESTUDIOS RELACIONADOS

La clasificación de comentarios de aplicaciones móviles puede dar lugar a muchos otros estudios relacionados. A continuación se listan algunas ideas planteadas durante la concepción o el desarrollo de este proyecto que también podrían resultar de enorme interés:

- Uso de clasificadores para identificar comentarios en los que se mencionan o comparan aplicaciones alternativas a aquella para la cual fue realizado el comentario. También podría ser posible clasificar estas comparativas entre aquellas que resultan positivas y las que resultan negativas para la aplicación comparada. Las aplicaciones posibles de este estudio son evidentes teniendo en cuenta el creciente interés de los últimos años en la inteligencia competitiva.[3]
- Uso de SVM de cálculo de regresiones (SVR) para detectar diferentes tendencias en los comentarios realizados con el objetivo, por ejemplo, de realizar una ordenación por relevancia según unos criterios determinados.

[3] www.kompyte.com

Referencias

A Tutorial on Support Vector Machines for Pattern Recognition

CHRISTOPHER J.C. BURGES

http://research.microsoft.com/pubs/67119/svmtutorial.pdf

Support vector machines for spam categorization (1999)

Harris Drucker , Donghui Wu , Vladimir N. Vapnik

http://www.site.uottawa.ca/~nat/Courses/NLP-Course/itnn_1999_09_1048.pdf

Mining Adverse Drug Reaction Signals from Social Media: Going Beyond Extraction

Apurv Patki, Abeed Sarker, Pranoti Pimpalkhute, Azadeh Nikfarjam, Rachel Ginn , Karen O'Connor, Karen Smith, Graciela Gonzalez

http://phenoday2014.bio-lark.org/pdf/2.pdf

Twitter Sentiment Classification using Distant Supervision

Alec Go, Richa Bhayani, Lei Huang

http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf

Text Categorization with Support Vector Machines: Learning with Many Relevant Features

Thorsten Joachims

http://www.cs.iastate.edu/~jtian/cs573/Papers/Joachims-ECML-98.pdf

Support-Vector Networks

Corinna Cortes, Vladimir Vapnik

http://homepages.rpi.edu/~bennek/class/mmld/papers/svn.pdf

A Training Algorithm for Optimal Margin Classifiers

Bernhard E. Boser , Isabelle M. Guyon , Vladimir N. Vapnik

http://www.svms.org/training/BOGV92.pdf

Multi-class Support Vector Machines

Jason Weston , Chris Watkins

http://ijcai.org/Past%20Proceedings/IJCAI-99%20VOL-2/PDF/011.pdf

LIBSVM: A Library for Support Vector Machines

Chih-Chung Chang and Chih-Jen Lin

http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf

SVM - Support Vector Machines

Modelos de aprendizaje supervisado con algoritmos de aprendizaje asociado que analizan datos y reconocen patrones, usados principalmente para realizar clasificaciones y análisis de regresión.

http://en.wikipedia.org/wiki/Support_vector_machine

LIBSVM

Librería implementada en C++ y de código abierto para construcción de modelos basados en Support Vector Machines. Dispone de interfaces y extensiones para gran cantidad de lenguajes de programación, entre ellos NodeJS.

http://www.csie.ntu.edu.tw/~cjlin/libsvm/

node-svm

Módulo para NodeJS basado en LIBSVM.

https://github.com/nicolaspanel/node-svm/tree/master

Freeling

Herramienta multiproposito de análisis de lenguaje de código abierto desarrollada por el TALP Research Center en la Universitat Politécnica de Catalunya. Dispone gran cantidad de herramientas de análisis (Splitter, POS Tagger, etc) para diversas lenguas.

http://nlp.lsi.upc.edu/freeling/

Javascript

Lenguaje de programación orientado a prototipos. Este lenguaje goza de gran popularidad actualmente, debido a que se trata de un lenguaje cómodo y que se puede usar tanto en el navegador como en el servidor web.

https://es.wikipedia.org/wiki/JavaScript

NodeJS

Entorno de programación en la capa del servidor orientado a eventos y basado en el motor V8 de Javascript.

https://es.wikipedia.org/wiki/Node.js

MongoDB

Sistema de base de datos NoSQL orientado a documentos implementado en C++ y que dispone de drivers oficiales para la mayoría de lenguajes de programación actuales.

https://es.wikipedia.org/wiki/MongoDB

The iPhone App Review

Revista online especializada en los análisis de aplicaciones móviles.

www.theiphoneappreview.com

PC Mag

Revista online sobre temas de informática y dispositivos móviles

http://www.pcmag.com

App Store

Tienda oficial de aplicaciones para iOS

http://www.apple.com/

Google Play

Tienda oficial de aplicaciones para Android

http://play.google.com

Blackberry World

Tienda de oficial de aplicaciones para Blackberry OS

https://appworld.blackberry.com/

Windows Phone Store

Tienda oficial de aplicaciones para windows phone

http://www.microsoftstore.com

Mevvy

Agregador de aplicaciones multiplataforma

http://www.mevvy.com/

AppBrain

Agregador de aplicaciones para Android

http://www.appbrain.com

Anexos

TECNOLOGÍAS Y LIBRERÍAS UTILIZADAS

LENGUAJE DE PROGRAMACIÓN

Todo el sistema desarrollado en este proyecto corre sobre la máquina de ejecución de Javascript V8, proporcionada por NodeJS. Hemos optado por el uso de este lenguaje de programación debido a que se trata de uno de los lenguajes de programación interpretados más eficientes y de mayor expansión en la actualidad.

IMPLEMENTACIÓN DE LOS SVM

Para la construcción de los SVM se ha optado por la utilización una implementación existente basada en código abierto. En particular se ha hecho uso de la librería de código abierto LIBSVM. Los factores considerados para llegar a esta decisión son los siguientes:

- LIBSVM es la librería más utilizada en la actualidad para la implementación de sistemas basados en SVM y su eficacia y estabilidad se encuentra ya demostrada en numerosas aplicaciones que hacen uso de la misma. (CONSULTAR LA TABLA ADJUNTA)
- En la actualidad, es la única librería que dispone de una interfaz ya implementada para su uso desde NodeJS. La interfaz en cuestión es “node-svm” y se encuentra disponible en el repositorio de paquetes NPM (EXPLICACIÓN DE QUE ES NPM EN EL PIE DE PAGINA).

SPLITTING Y TOKENIZACIÓN

Para la identificación de las oraciones y la tokenización de sus elementos ha sido utilizada la librería de análisis multipropósito Freeling. Se ha decidido utilizar dicha plataforma en lugar de realizar una implementación propia debido a que esta permite un gran abanico de posibilidades que, si bien no han sido completamente aprovechadas para las pruebas realizadas, podrían ser de gran utilidad en extensiones futuras; como ya se comenta en el apartado correspondiente.

BASE DE DATOS

Para la persistencia de los datos, se ha optado por el uso de una base de datos NoSQL, MongoDB. Se ha decidido hacer uso de este tipo de bases de datos debido a que sus colecciones de documentos no tienen por qué tener una estructura fija y es muy fácil añadir a posteriori nuevos campos. Además, trabaja especialmente bien para cantidades elevadas de documentos y su integración con NodeJS es muy sencilla.

TABLA COMPARATIVA DE LAS LIBRERÍAS DE SVM MÁS UTILIZADAS[4]

A continuación se adjuntan las 9 primeras filas del ranking de uso de las diferentes librerías de SVM más utilizadas elaborado por el sitio svms.org. Esta ordenación atiende al número de publicaciones o webs en las que se encuentran mencionadas cada una de las diferentes librerías.

Google Directory Rank	Google PageRank	Backward Links	Web	Scholar	Software	Details
2	(6/10)	~92	~50,900	~3,580	LIBSVM	Both C++ and Java sources
1	(6/10)	~68	~19,600	~1,450	SVMlight	Written in C
-	(5/10)	3	~753	~488	SVMTorch	C++
-	(5/10)	~9	~579	~232	mySVM	C++
5	(0/10)	0	~308	~179	TinySVM	Written in C++ with OO style
-	(5/10)	3	~142	~152	BSVM	C and C++
3	(6/10)	6	~309	~102	SvmFu	written in C++
-	(5/10)	2	~93	~551	OSU SVM Classifier Matlab Toolbox	Matlab Toolbox
-	(5/10)	8	~314	~221	Gist	written in ANSI C

4 Fuente: http://www.svms.org/software.html

APPNALYZE DOWNLOADER

En este anexo se explicarán los detalles más relevantes de la herramienta auxiliar desarrollada para poder realizar correctamente la extracción y marcado de los diferentes comentarios de de la tienda de aplicaciones móviles “Google Play”.

EXTRACCIÓN DE LOS COMENTARIOS

Pese a que pueda resultar una funcionalidad de lo más básica, a Septiembre de 2014 aún no existe una API pública para descargar los diferentes comentarios de una determinada aplicación en la tienda de aplicaciones “Google Play”. Es por esto que fue necesario recurrir a métodos de ingeniería inversa para realizar la satisfactoria extracción de los comentarios de dicha tienda.

Después de analizar las comunicaciones realizadas por la versión web de “Google Play” utilizando el debugger integrado en el navegador web Google Chrome, fui capaz de concluir que los comentarios de una determinada aplicación se descargan dinámicamente usando AJAX.

Más específicamente, los comentarios se descargan realizando peticiones al método HTTP POST / store/getreviews. Dicho método acepta los siguientes parámetros : reviewType, pageNum, id, reviewSortOrder, y xhr. De estos parámetros, los únicos con relevancia para nuestra aplicación son “pageNum”, que especifica el índice de la página de comentarios a descargar; e “id”, que especifica la el identificador de la aplicación en particular.
El identificador de una aplicación de Google Play es el nombre del paquete principal de la misma. Una de las maneras más sencillas de conocer este identificador es copiando el el parámetro HTTP GET “id” de la url de la página de detalles de la misma en el sitio web de Google Play.

También pude comprobar que configuración regional y las preferencias del usuario se comunican al servicio web mediante cookies. Los cookies usados para este efecto son “NID” y “PLAY_PREF”, siendo el primero el que identifica al usuario y el segundo el que almacena la configuración de la web. Cabe aclarar que el valor de dichos cookies se encuentra codificado y su interpretación queda fuera del abasto de este proyecto. Es posible obtener valores válidos de estos parámetros abriendo el sitio web de “Google Play” con un navegador web, abriendo las herramientas para desarrolladores y copiando los valores de las cookies asociadas al sitio.

La respuesta a dicho servicio web es el fragmento de código HTML por el cual será reemplazada la página de comentarios actual en el navegador. Puesto que dicha respuesta se envía mediante texto plano, los caracteres conflictivos HTML se encuentran escapados.

El texto de un comentario se encuentra dentro de una etiqueta de tipo “div” y con la clase de CSS “review-body”. El título se puede identificar fácilmente, pues está contenido de una etiqueta de tipo “span” y clase “review-title”. Otro dato importante a extraer es el identificador de comentario, que se encuentra en el parámetro “data-reviewid” de las etiquetas de tipo “div” y clase “review-header” que engloban el resto de información del comentario.

A vista de estos datos, desarrollé una aplicación que simula las peticiones del navegador Google Chrome y envía las cookies de un usuario anónimo accediendo al sitio web desde España. Dichas peticiones se realizan a intervalos de tiempo aleatorios entre 1 y 4 segundos con el fin de evitar ser detectados por los sistemas de protección contra robots de los servidores de Google. Asimismo analiza las repuestas y les proporciona un formato adecuado para que puedan ser analizados en pasos posteriores.

PROCESADO Y ENRIQUECIDO DE LOS COMENTARIOS

Como ya se comenta en el en este documento, para poder parametrizar y enriquecer los documentos hemos hecho uso de la herramienta de análisis de texto multipropósito FreeLing. Con el fin de automatizar completamente el proceso de generación de los diferentes ejemplos, Appnalizer Downloader también es capaz de ejecutar y comunicarse con procesos ‘analyzer’ de Freeling.

Una vez han sido descargados todos los comentarios de una aplicación, Appnalizer Downloader arranca una instancia del programa ‘analyzer’ en modo servidor con la configuración por defecto para la lengua castellana y analiza cada uno de los comentarios ejecutando instancias de ‘analyze’ en modo cliente. El resultado del análisis de estos documentos se obtiene creando una tubería con la salida estándar de dichos procesos.

Una vez procesados, los comentarios estarán divididos por oraciones, tokenizados y enriquecidos con la categoría dentro de la oración (“tag”) y el lema (“lemma”) de cada uno de los tokens que componen.

Cabe aclarar que la tokenización realizada por Freeling difiere de una tokenización automática tradicional, es decir, separar las palabras por espacios, saltos de línea y tabuladores; en que las palabras compuestas se agrupan como una única entidad. De este modo, construcciones como a “a menudo” o nombres como “Vía Láctea” serán considerados un único token. Ese hecho se suma

a los anteriormente mencionados para permitir que nuestros documentos de entrada sean lo más ricos e informados posible.

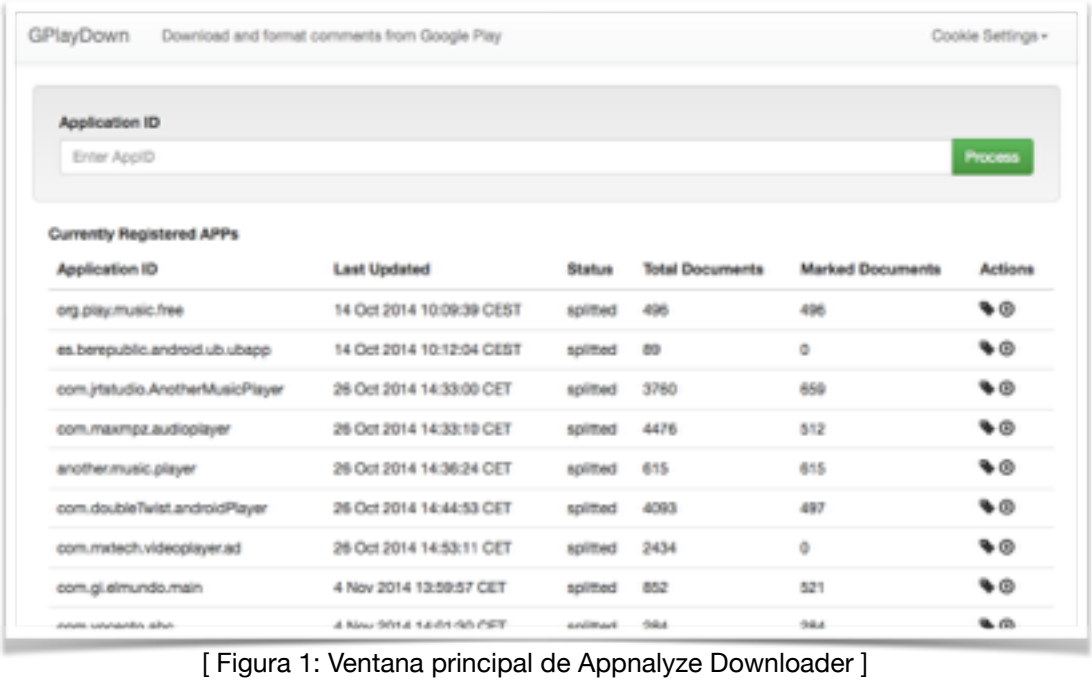
INTERFAZ GRÁFICA DE USUARIO (GUI)

Para añadir nuevas tareas a Appnalizer Downloader se optó por la realización de una simple interfaz web, que posteriormente fue enriquecida con diferentes funciones adicionales como son el reprocesado de comentarios o la visualización de estadísticas.

FUNCIONES BÁSICAS DE LA GUI

Descargar comentarios

Para programar la descarga y procesado de los comentario de una nueva aplicación lo único que es necesario hacer es introducir el identificador de la aplicación en el campo de texto “Application ID” y pulsar el botón process. A continuación la aplicación será añadida a la lista “Currently Registered APPs” en estado “Downloading”.



[Figura 1: Ventana principal de Appnalyze Downloader]

Configuración de cookies

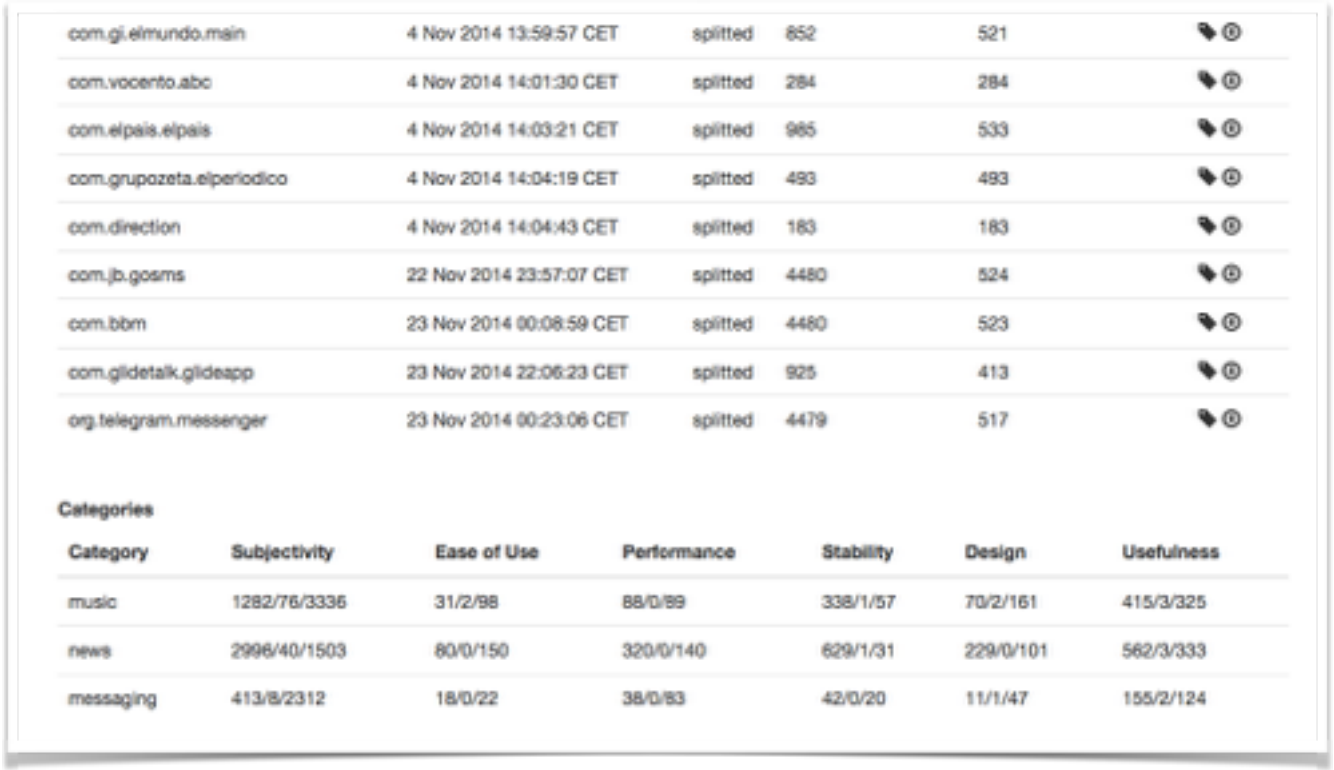
En la esquina superior derecha de la pantalla se encuentra el botón “Cookie Settings”. Pulsando dicho botón se abrirá un formulario en el que se podrá sobreescribir la configuración de los “cookies” de identificación del usuario que serán usados por el descargador la próxima vez que se programe la descarga de de comentarios de una aplicación. Cambiando esta configuración será posible, por ejemplo, descargar comentarios con una configuración regional diferente.

Visualizar aplicaciones procesadas:

En la lista “Currently Registered APPs” se pueden visualizar las distintas aplicaciones que se encuentran registradas en el sistema junto con la fecha de última actualización de los comentarios, el estado dentro de la cola de procesos, el número total de comentarios y el número de comentarios que han sido marcados como ejemplos.

Reprocesar los comentarios de una aplicación

Es posible forzar la actualización y reprocesado de los comentarios de una aplicación haciendo click en el segundo botón de la columna “actions” de la lista de aplicaciones procesadas.



com.gi.elmundo.main	4 Nov 2014 13:59:57 CET	splitted	852	521	
com.vocento.abc	4 Nov 2014 14:01:30 CET	splitted	284	284	
com.elpais.elpais	4 Nov 2014 14:03:21 CET	splitted	985	533	
com.grupozeta.elperiodico	4 Nov 2014 14:04:19 CET	splitted	493	493	
com.direction	4 Nov 2014 14:04:43 CET	splitted	183	183	
com.jb.gosms	22 Nov 2014 23:57:07 CET	splitted	4480	524	
com.bbm	23 Nov 2014 00:08:59 CET	splitted	4480	523	
com.glidetalk.glideapp	23 Nov 2014 22:06:23 CET	splitted	925	413	
org.telegram.messenger	23 Nov 2014 00:23:06 CET	splitted	4479	517	

Categories

Category	Subjectivity	Ease of Use	Performance	Stability	Design	Usefulness
music	1282/76/3336	31/2/98	88/0/89	338/1/57	70/2/161	415/3/325
news	2996/40/1503	80/0/150	320/0/140	629/1/31	229/0/101	562/3/333
messaging	413/8/2312	18/0/22	38/0/83	42/0/20	11/1/47	155/2/124

[Figura 2: Lista de categorías de la pantalla principal]

Visualizar estadísticas sobre las categorías

En la lista “Categories” es posible visualizar el número de ejemplos clasificados para cada categoría de aplicación. Los datos se muestran desglosados por característica. Para cada característica se muestra el número de ejemplos de polaridad negativa, neutra y positiva separados por barras laterales.

Ejecutar Appnalize Marker

Es posible ejecutar el sistema de marcado manual de ejemplos para los comentarios de una aplicación haciendo click en el primer botón de la columna “actions” de la lista de aplicaciones proceadas.

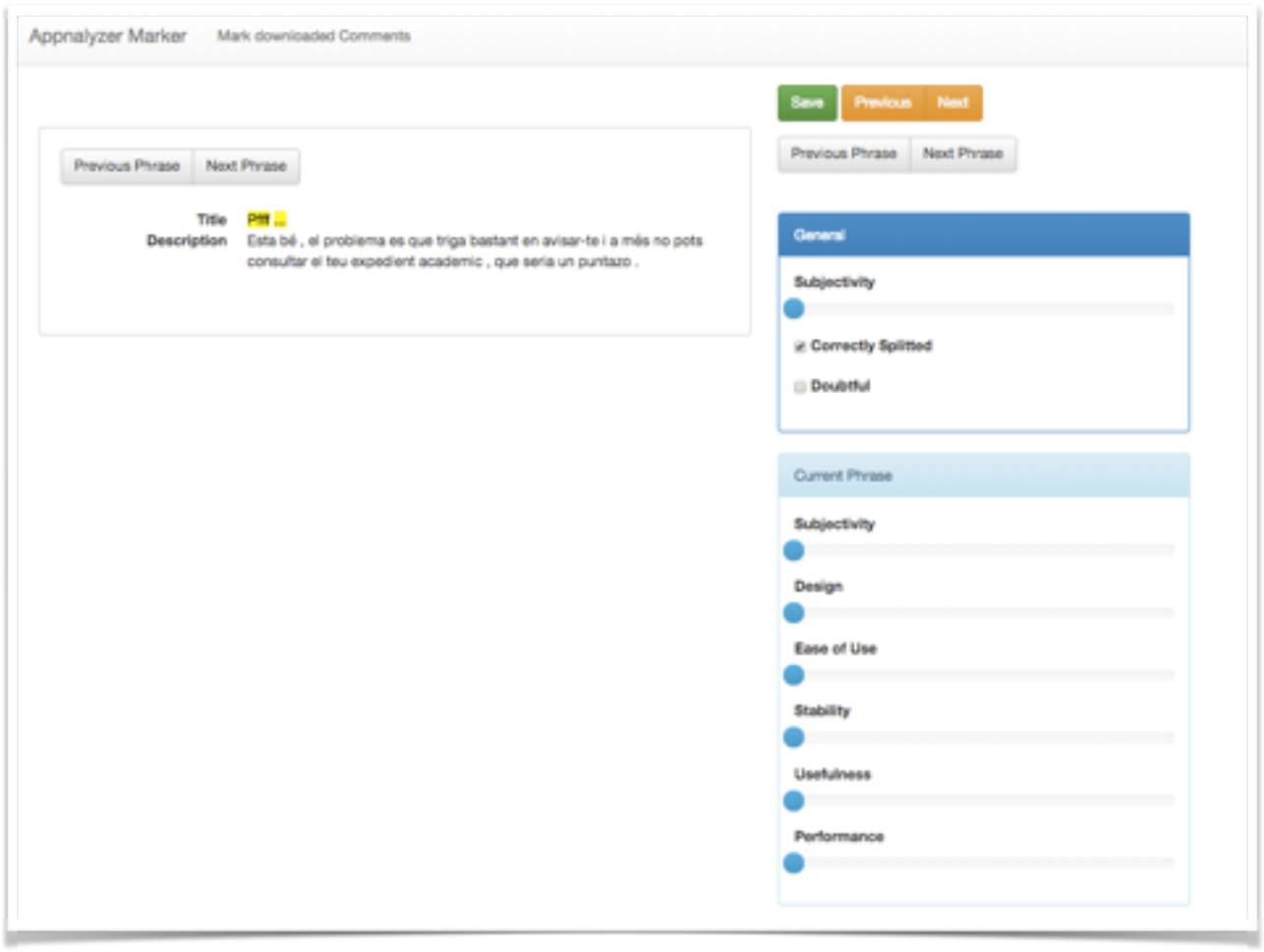
APPNALYZE MARKER

Junto con el sistema de obtención y enriquecimiento de de los documentos de opinión, se ha desarrollado un sistema de marcado manual de documentos. El sistema ha sido desarrollado específicamente para este proyecto y es capaz de interpretar el formato interno de la aplicación, separando cada comentario en las diferentes oraciones que lo componen.

La motivación principal del desarrollo de este sistema en lugar de usar una de las aplicaciones gratuitas que se pueden encontrar es el hecho de que ninguna de las mencionadas aplicaciones dispone de una versión con interfaz adaptada a dispositivos móviles.

Gracias a este módulo, me ha sido posible realizar el marcado de los documentos durante los largos viajes diarios en transporte público, momento durante el cual no tengo un ordenador a mi disposición.

Otra ventaja fundamental de este sistema es el hecho de que no es necesario instalar ninguna aplicación en cada uno de los sistemas desde los cuales se desean marcar documentos. Únicamente es necesario un navegador web para añadir los datos de marcado a los documentos. Además no es necesario copiar ni sincronizar los comentarios marcados, pues estos se encuentran centralizados en el servidor central.



[Figura 3 : Ventana principal de Appnalyze Marker]

Como se puede apreciar en la figura, la interfaz es sencilla, concisa y clara. Los comentarios se dividen por oraciones, estando la activa claramente subrayada en un fondo amarillo.

La introducción de la polaridad de cada característica se realiza mediante un slider dentro la de la sección “Current Phrase”. Moviendo el slider es posible ver el valor seleccionado en cada momento mediante un overlay.

La posición más a la izquierda del slider (valor 0) representa la no subjetividad. Los valores 1 a 5 representan la subjetividad de la oración, siendo 1 el valor más negativo y 5 el valor más positivo. El valor 3 se usa para representar la neutralidad de la oración, es decir, cuando se trata de una oración subjetiva pero no es posible determinar su polaridad.

La sección “General” se usa para especificar cierta información común a todo el documento. Dentro de este apartado podemos encontrar un slider para especificar la subjetividad del comentario en su conjunto, dato que hemos decidido no utilizar en este proyecto, y dos checkboxes: correctly splitted y doubtful. El primer checkbox se encuentra marcado por defecto y se usa para confirmar que el splitting del documento ha sido realizado correctamente. El segundo se usa para alertar de que un documento es excesivamente dudoso, es decir, no es posible determinar ni siquiera si se trata de un documento de opinión o no.

Además de estos selectores, se pueden encontrar los botones “Previous”, “Next”, “Previous Phrase”, “Next Phrase” y “Save”. Los botones “Previous Phrase” y “Next Phrase” se usan para cambiar la frase activa dentro del comentario actual. Los botones “Previous” y “Next” sirven para substituir el comentario actual. Cada vez que el comentario actual es substituido o se pulsa el botón “Save”, la información se sincroniza con el servidor.

El marcado de los documentos puede realizarse estando sin conexión, ya que todos los documentos de la aplicación se descargan en la carga de la página. Cuando el usuario vuelve a tener conexión simplemente debe pulsar el botón “Save” para que la información de las marcas sea sincronizada con el servidor.

APPNALYZE GRAPHER

Para la facilitar la representación de los valores obtenidos para las diferentes pruebas y generar cada una de las gráficas incluidas en esta memoria ha sido desarrollada la sencilla aplicación web Appnalyze Grapher.
El dibujado de la gráficas es realizado por medio la librería de javascript highcharts, incluyendo el módulo adicional de exportación. Este módulo permite por tanto que las gráficas generadas con Appnalyze Grapher sean cómodamente exportables a los formatos PNG, JPEG, PDF y SVG.
Appnalyze Grapher consulta la base de datos para los parámetros fijados en su formulario y realiza una representación gráfica de la variable seleccionada en función del valor de la variable C para los diferentes kernels estudiados.

A continuación se listan las variables que pueden ser representadas gracias a Appnalyze Grapher:

Variable	Valor	Limiaciones
F	fscore	
Accuracy	accuracy	Document class = “any”
Recall	recall	Document Class = 0 / 1 / 2 / 3
Precision	precision	Document Class = 0 / 1 / 2 / 3
Predicciones Correctas	tp	Document Class = 0 / 1 / 2 / 3 nfold = 0 / 1 / 2 / 3 / 4
Predicciones Totales	predictions	Document Class = 0 / 1 / 2 / 3 nfold = 0 / 1 / 2 / 3 / 4
Predicciones Esperadas	expected	Document Class = 0 / 1 / 2 / 3 nfold = 0 / 1 / 2 / 3 / 4
Tiempo de test	testing_time	Sólo para pruebas en las que no se ha realizado cross validation
Tiempo de entrenamiento	training_time	Sólo para pruebas en las que no se ha realizado cross validation

Los diferentes parámetros de filtrado junto con sus valores posibles se listan a continuación. Si no se introduce ningún valor en el campo, no se realizará ningún filtrado por dicho parámetro.

Parámetro	Valores posibles	Explicación
Collection	“tests”, “tests_old”	Colección de la cual serán leídos los valores. Por defecto “tests”.
Document Subtype	“subjectivity”, “design”, “performance”, “ease”, “usefulness”, “stability”	Característica clasificada
Document Gravity	“any”, “positive”, ”negative”, “positive-neutral”, “negative-neutral”, “multiclass”	Polaridad clasificada
Document Class	“any”, 0, 1, 2, 3	Clase del documento. Cuando se desean obtener los condicionados de recall y precision.
Test Name	Cualquier texto	Nombre del test consultado
Corpus	“training”, “test”	Corpus evaluado
Multiclass Mode*	0, 1, 2, 3	Para el caso multiclase, clase asignada al conjunto de intersección
NFold	“average”, 0, 1, 2, 3, 4	Partición consultada
Variable	Las listadas en la tabla anterior	Variable representada

SISTEMA DE TEST DE APPNALYZE

INTRODUCCIÓN

En un proyecto como Appnalyze, disponer de un sistema de pruebas fiable y automatizado es de vital importancia. Con miles de pruebas a realizar y procesar, es de vital importancia que los resultados obtenidos se encuentren lo mejor clasificados posible.
Otro factor que hemos de tener en cuenta es la comodidad a la hora de utilizar y representar estos resultados. Es importante además garantizar la persistencia de la mayor cantidad de información posible, con el fin de que los resultados obtenidos puedan ser confirmados con posterioridad.

PERSISTENCIA

Para gestionar la persistencia, hemos optado por combinar el uso de un sistema de ficheros convencional con una base de datos MongoDB.
Hemos usado la base de datos para almacenar y ordenar los resultados de las pruebas. De este modo la obtención de los valores para la representación gráfica de los diferentes parámetros de calidad de los clasificadores es mucho más cómoda y rápida. Además resulta muy simple cambiar el parámetro a representar, las series que se representarán o el parámetro sobre el que se itera.
Pese a que para tareas de representación de resultados la base de datos es una herramienta muy útil, usarla para almacenar todos los documentos intermedios es inviable. Por un lado, el volumen de datos es excesivamente elevado. Por otro, las librerías de acceso y la comunicación con el sistema de gestión de bases de datos representa un overhead innecesario para los procesos de generación y testado de los SVMs.
Es por esto que hemos optado por almacenar todos los documentos intermedios utilizados por los SVMs en el sistema de ficheros. Estos documentos solo serán utilizados durante la fase de generación y evaluación de los SVMs y pasarán a ser innecesarios en las fases de sintetizado de los resultados obtenidos. Pese a esto, hemos decidido no eliminar dichos archivos con el fin de que sea posible volver comprobar la veracidad de todos los resultados presentados en este documento con posterioridad.

PROCESO DE TEST

Todas las pruebas realizadas para este proyecto han sido realizadas siguiendo un proceso estandarizado. Para ayudar en este proceso se han implementado tres aplicaciones especializadas en las tareas de preparación y planificación (SVM Generator Scheduler); construcción y evaluación (SVM Generator); y sintetización y almacenado (Database Filler).
Las razónes principales por la cuales se ha decidido realizar 3 aplicaciones diferenciadas en lugar de una única aplicación que resuma todo el proceso son la estabilidad y la eficiencia. Teniendo en cuenta que este proyecto ha sido llevado a cabo en máquinas personales, los problemas de exceso

de memoria o número de procesos son muy probables. Utilizando el sistema planteado, una excepción por exceso de memoria sólo afectará a la subtarea en la que se ha disparado. Además, elegir el número de tareas que se realizarán en paralelo es tan simple como modificar un parámetro de configuración.

DESCRIPCIÓN DEL PROCESO

Para cada una de las diferentes pruebas que se han realizado, se ha generado un archivo de configuración. Esta configuración es leída por el SVM Generator Scheduler para planificar las diferentes tareas de generación y testeado en las que se descompone la mencionada prueba y ejecuta una instancia de SVM Generator para cada una de ellas.

Los resultados de cada una de las diferentes pruebas se almacenan en archivos separados, identificados por la característica analizada, la polaridad y el timestamp del momento en el que fueron realizados. Dentro de estos archivos se informa del resto de parámetros usados para la prueba en particular, así como el tiempo que se ha tardado en realizar dicho test.

Por limitaciones de la librería utilizada, no es posible obtener los resultados de la ejecución del corpus de test para cada uno de los parámetros cuando se realizan pruebas que requieren la generación de más de un SVM. Esto es, cuando se utiliza cross-validation o cuando se utiliza un rango de valores para alguno de los parámetros. Es por esto que cuando se han realizado pruebas con corpus de test, no se han utilizado rangos para ninguno de los parámetros en SVM Generator.

SVM GENERATOR SCHEDULER

La primera aplicación que ejecutamos al realizar un test es “svm_generator_scheduler”. Esta aplicación se encarga de leer los parámetros sobre los que se iterará en dicho test y prepara los corpus de aprendizaje y test que serán usados para generar y validar los SVMs.

FUNCIONALIDADES

- Descarga y procesado de los documentos marcados: Descargar los documentos marcados mediante la herramienta Appnalyze Marker. Los documentos que cumplen las opciones especificadas en la configuración son almacenados en el archivo “applications.json”.
- Factorización y transformación de los documentos al dominio vectorial: A partir de una lista de documentos en el formato de Appnalyze Marker, ser capaz de parametrizar los diferentes documentos basándose en los tokens que los conforman. Para esto genera un archivo de traducción de tokens al dominio vectorial (El archivo “mapping_values”).
- Reducción de dimensiones en el dominio vectorial: Reducir las dimensiones del dominio vectorial atendiendo a, por ejemplo, la frecuencia de aparición de cada token.
- Generación de los corpus de training y test: A partir de las representaciones de los documentos en el dominio vectorial y de los valores de polaridad para cada característica analizada, generar archivos que serán usados como corpus de test y aprendizaje por SVM Generator.
- Reducción y equilibrado de los corpus: Eliminar ejemplos duplicados o equilibrar el número de ejemplos para cada una de las clases que serán identificadas por los SVMs.
- Planificación y ejecución de los diferentes test: Ejecutar secuencial o paralelamente procesos de “SVM Generator” con cada una de las configuraciones deseadas. El número de ejecuciones paralelas puede ser también configurado.

CONFIGURACIÓN:

La configuración del SVM Generator Scheduler se establece mediante un simple archivo en formato javascript en el cual se exportan 2 variables: OPTIONS_ARRAY y TEST_ARRAY.

La variable OPTIONS_ARRAY contiene una lista de características sobre las cuales se realizará el análisis junto con la polaridad esperada. El nombre de la combinación de características y polaridad se especifica mediante el campo “subtype”.

La variable TEST_ARRAY contiene las diferentes configuraciones de las pruebas a realizar. . Cabe aclarar que esta variable se comporta como una tubería, es decir, cada una de las listas de opciones se ejecuta inmediatamente después de la anterior. Esto permite crear procesos en los cuales las entradas de la operación especificada por una configuración depende obligatoriamente del resultado de la operación especificada por la configuración anterior.

A continuación se listan las diferentes opciones que pueden ser establecidas para cada una de las configuraciones de TEST_ARRAY:

- categories : Lista de los dominios de comentarios a partir de los cuales se extraerán los documentos.
- reduce_examples : Indica si se desea que el generador de corpus reduzca los mismos eliminando los ejemplos duplicados. Su valor por defecto es “true”.
- folder: Directorio de trabajo utilizado por la configuración.
- parameters: Lista de parámetros estáticos que serán pasados al SVM Generator.
- iterate_parameter : Lista de parámetros del SVM Generator que serán iterados. Se ejecutará una instancia diferente de SVM Generador para cada combinación de valores en este campo.
- generate_svm: Indica si se desea que se generen SVMs. Su valor por defecto es “true”.
- attribute_map_folder : Directorio del cual se leerá la tabla de traducciones de cada token en su dimensión asociada dentro del dominio del SVM generado. Por defecto, su valor es el mismo que el de la opción “folder”.
- read_attribute_map”: Indica si se desea generar una nueva tabla de traducciones de tokens o si se desea usar la que se encuentre el el directorio especificado en “attribute_map_folder”. Su valor por defecto es “false”.
- read_from_database : Indica si se desea que los documentos sean cargados de la base de datos o leídos del archivo “applications.json” en el directorio especificado en “folder”. En el caso de que los documentos sean leídos de la base de datos, se generará o sobreescribirá el archivo “applications.json”. Su valor por defecto es “true”.
- limit_attributes_to : Indica el número de tokens al cual serán restringidos los mapas de atributos. Todos los tokens que no cumplen con ninguna de las clasificaciones serán agrupados bajo el parámetro “otros”. De este modo, el número de dimensiones en el espacio vectorial transformado será “limit_attributes_to” + 1. Por defecto no se limita el número de atributos.

- limit_attributes_policy: Indica la política de selección de tokens cuando se establece una limitación en el número de los mismos por medio de “limit_attributes_to”. Este parámetro puede tomar valores de 0 a 1 e indica el porcentaje de tokens no seleccionados con más frecuencia que aquellos seleccionados. Así, para seleccionar los tokens con mayor frecuencia “limit_attributes_policy” tomará el valor 0 y para seleccionar aquellos con menor frecuencia se asignará el valor 1. Su valor por defecto es 0, es decir, seleccionar siempre los más frecuentes.

SVM GENERATOR

El SVM Generator es la aplicación encargada de la construcción y evaluación de un SVM para unos parámetros determinados, es decir, ejecuta una prueba aislada. Todos los resultados obtenidos son escritos posteriormente a un archivo de reporte. Todos los archivos intermedios son almacenados a su vez en ficheros para permitir su posterior revisión.
SVM Generator es capaz de realizar dos tipos de pruebas: simples y de cross-validation. La realización de una u otra dependerá de si se dispone de dos corpus diferenciados para aprendizaje y para test o si por contra sólo se dispone de un corpus común.
Para el caso de las pruebas simples, se usa todo el corpus de entrenamiento para generar un modelo y para la evaluación se toma el corpus de test. Tanto el resultado de la evaluación del corpus de entrenamiento como del corpus de test son almacenados.
En el caso del modo de cross-validation, el corpus de entrada se particiona en 5 subconjuntos y se realizan 5 pruebas sucesivas tomando 4 subconjuntos como entrenamiento y el restante como test, presentándose los valores obtenidos para cada una de las pruebas así como el valor medio de cada uno de los parámetros de evaluación.

CONFIGURACIÓN

SVM Generator no utiliza archivos de configuración sino que recibe toda su configuración mediante parámetros en la línea de comandos en formato '--parámetro="valor"'. Los valores recibidos como parámetro serán redigidos a la librería 'libsvm", es por esto que podemos decir que SVM Generator se comporta como un wrapper de la misma. Las opciones más relevantes son las siguientes:

- --folder: Directorio donde se encuentra el corpus de aprendizaje.
- --test_folder: Directorio donde se encuentra el corpus de test.
- --svm_kernel: Tipo de kernel usado para el SVM. Puede tomar los siguientes valores: Puede tomar los siguientes valores: LINEAR, POLY, RBF y SIGMOID. El valor por defecto es POLY (polinómico).
- --svm_type: Tipo del SVM generado. El único valor soportado actualmente es CSVC, es decir, SVM clasificador.
- --subtype: Subtipo del análisis. El valor se compone por la característica a analizar seguida de una barra baja y la polaridad. Este valor se usará para seleccionar los archivos a analizar de los directorios de trabajo.
- --eps: Valor del parámetro epsilon. El valor por defecto es 1e-2.
- --reduce / --no-reduce: Usar o no usar el algoritmo de reducción de dimensiones proporcionado por la librería "node-svm". Por defecto no se reducen las dimensiones..
- --C: Parámetro de coste del clasificador CSVC. Por defecto itera sobre los valores 0.03125, 0.125, 0.5, 2 y 8.

- --gamma: Parámetro del kernel. Sólo se considerará su valor para los kernels POLY, RBF y SIGMOID. Por defecto itera sobre los valores 0.03125, 0.125, 0.5, 2 y 8.
- --r: Parámetro del kernel. Sólo se considerará su valor para los kernels POLY y SIGMOID. Por defecto itera sobre los valores 0.03125, 0.125, 0.5, 2 y 8.
- --degree: Parámetro del kernel. Sólo se considerará su valor para el kernel POLY. Por defecto itera sobre los valores 0.03125, 0.125, 0.5, 2 y 8.
- --nFold: Número de divisiones si se desea usar una prueba de cross-validation. Por defecto 5. Si no se desea realizar cross-validation, el parámetro deberá tomar valor 1.
- --normalize / --no-normalize: Especifica si se desean normalizar los ejemplos de los corpus. Por defecto no se normalizan los corpus.

DATABASE FILLER

La aplicación Database Filler es la encargada de almacenar todos los resultados obtenidos por SVM Generator en la base de datos. Los resultados son leídos secuencialmente y clasificados atendiendo al tipo de prueba realizada, cardinalidad de las clases, polaridad, característica analizada, etcétera. Además de esto, Database Filler proporciona una funcionalidad adicional: la simulación de clasificadores multi-clase a partir de las predicciones realizadas mediante clasificadores binarios. Esto se realiza obteniendo los conjuntos de intersección de dos clasificadores binarios y asignando una clase diferente a los documentos de dicho subconjunto. Database Filler genera los valores de precisión, recall, accuracy y F del clasificador para cada una de las posibles clases asignadas al conjunto de intersección.

CONFIGURACIÓN

Al igual que SVM Generator, Database Filler no utiliza archivos de configuración sino que recibe toda su configuración mediante parámetros en la línea de comandos en formato '--parámetro="valor"'. Los parámetros de configuración son los siguientes:

- --folder: Directorio donde se encuentran los archivos de resultado de las pruebas.
- --test_folder: Directorio donde se encuentran los archivos intermedios resultado de la evaluación del corpus de test.

- --training : Lista de categorías de comentarios que conforman el corpus de aprendizaje. Por defecto es “music,messaging,news”.
- --test: Lista de categorías de comentarios que conforman el corpus de test. Por defecto es vacío.
- --tested / --no-tested : Especifica si se ha utilizado un corpus de test. Por defecto es no testeado.
- --test_collection : Colección de la base de datos en la cual serán guardados los resultados.
- --emulate-multiclass / --no-emulate-multiclass : Simular un clasificador multiclase a partir de dos clasificadores binarios.
- --svm_type : Tipo de SVM utilizado. Por defecto es “POLY”.
- --test_name : Nombre del test realizado. Por defecto es “default”.
- --balanced / --no-balanced : Especifica si se ha utilizado un corpus balanceado o no. Por defecto es cierto.
- --reduced / --no-reduced : Especifica si se ha reducido el número de dimensiones del espacio utilizado por el clasificador o no. Por defecto es falso.

ESPECIFICACIONES DEL EQUIPO DE PRUEBAS

A continuación se listan las características técnicas de equipos utilizados para pruebas y base de datos. Todos los tiempos de proceso listados en esta memoria son relativos al ordenador personal de pruebas detallado a continuación:

Equipo de Pruebas:

Sistema Operativo:	Arch Linux x86_64
Procesador:	Intel i5 2500K de 4 núcleos a 3.4GHz
RAM:	2x 4GB DDR3 a 1600MHz CL9
Disco Duro Principal:	SSD Samsung 830 de 128GB
Disco Duro Secundario:	HDD Seagate Barracuda de 1TB a 7200rpm
Swap:	8 GB

Servidor de Front y Base de Datos:

Sistema Operativo:	Ubuntu Server 14.04_2 LTS x86_64
Procesador:	Intel Atom N2800 de 2 núcleos a 1.8 GHz
RAM:	1x 2GB DDR3 a 1066MHz
Disco Duro Principal:	HDD TOSHIBA de 500GB a 7200rpm
Swap:	5 GB
Hosting:	Kimsufi (OVH)